Asservissement visuel direct et navigation utilisant un capteur catadioptrique

Bertrand Delabarre

Asservissement visuel direct et navigation utilisant un capteur catadioptrique

Rapport de Stage

Auteur :
Bertrand Delabarre

Superviseur :
Eric Marchand

Master recherche informatique

Promotion DIIC3 - Imagerie numérique

Université de Rennes 1 - ESIR

IRISA, INRIA Rennes - Bretagne Atlantique

Projet LAGADIC

Résumé

Dans ce rapport sont traités les problèmes d'asservissement visuel et de navigation, des sujets très actifs dans le domaine de la vision robotique. Beaucoup de travaux existent, principalement utilisant des techniques dites géométriques, c'est à dire se basant sur des primitives géométriques. Ces techniques nécessitant une phase de suivi ou d'extraction elles n'utilisent pas toute l'information de l'image. C'est pour cela qu'ont été développées récemment des techniques, dites directes, qui utilisent toute l'image sans suivre ou extraire de primitives géométriques. Une de ces techniques se base sur l'information mutuelle ([22]) des images et a été développée pour des images acquises par des caméras *perspective* ([7]). Cette méthode a pour principal avantage le fait de posséder les propriétés de robustesse dues à l'information mutuelle, notamment vis à vis des changements d'illumination, des occlusions ou de la multimodalité. La limite de cette approche réside dans le fait que des images de ce type amènent peu d'information du fait de leur champ de vision réduit.

Pour dépasser cette limite, une solution possible est de l'adapter à des images omnidirectionnelles qui apportent une information sur la scène à 360 $^\circ$. Dans ce rapport seront rappelés dans un premier temps les travaux existants dans le domaine de l'asservissement visuel avec des caméras perspective. Dans un second temps les méthodes d'asservissement visuel omnidirectionnel serons introduites avant que notre méthode soit développée. Cette dernière sera par la suite validée par des tests expérimentaux réalisés sur un robot cartésien à six degrés de liberté. Enfin dans une troisième partie un rappel des travaux sur la navigation de robot sera fait avant que ne soit montré comment cet asservissement peut être adapté à la navigation. La méthode proposée sera ici aussi validée avec cette fois des expériences successives en simulation, sur le robot cartésien à six degrés de liberté avant que ne soit détaillé un test grandeur nature sur un robot mobile non holonome.

Table des matières

Introduction

Chez l'homme, se déplacer dans l'environnement qui l'entoure est une tâche naturelle. Cette navigation est possible car il perçoit cet environnement qu'il entoure, principalement grâce à la vision. La technique permettant de mieux en mieux aux robots d'appréhender cet environnement qui les entoure, c'est en toute logique que des travaux récents ont cherché à leur adapter ce modèle de navigation basé sur la vision. Mais il est erroné de penser qu'il suffit de voir pour se déplacer facilement. En effet la vision chez l'homme se résume à bien plus que ses yeux. C'est le cerveau qui effectue la grande majorité du travail. Pour ce faire il se base sur des années d'expérience du monde qui nous entoure afin de se créer une banque d'*a priori* qui lui permettent d'interpréter ce qui est capturé par les yeux. De plus, le cerveau connaît les limites du corps et les intègre naturellement à ses réflexions ce qui crée des mouvements toujours naturels. Le robot aussi a des contraintes, encore plus que le corps humain, mais il ne peut les prendre tout seul en compte. Dans ce contexte il est facilement compréhensible que les problèmes de vision robotique soient des sujets de recherche très actifs.
Le processus permettant de contrôler un robot en se basant sur l'information fournie par une ou plusieurs caméras est appelé asservissement visuel. Pour effectuer ce contrôle beaucoup de méthodes ont été proposées en se basant sur des informations visuelles géométriques extraites des images acquises. Mais du fait de cette étape d'extraction d'informations visuelles ce n'est pas l'image dans sa totalité mais seulement une partie de cette dernière qui est utilisée. Cela entraîne une complexification du problème car il faut détecter et suivre ces primitives de manière efficace et rapide pour que l'asservissement visuel qui en découle soit de bonne qualité. Pour tenter de palier à ces contraintes d'autres techniques dites directes ou denses ont alors été proposées, utilisant l'image dans son ensemble pour réaliser la tâche d'asservissement visuel. Ces techniques se basent sur l'information photométrique contenue dans l'image et considèrent donc toute l'information de l'image. Cependant pour pouvoir se servir de ce nouveau type d'information le processus doit alors être revu et de nouvelles méthodes doivent être définies.

Le but de ce stage est donc double. Dans un premier temps il s'agit de définir et de tester une méthode d'asservissement visuel se basant sur l'information mutuelle d'images omnidirectionnelles. Une fois cela effectué il s'agit alors de définir un *framework* de navigation se basant sur cet asservissement visuel et de le valider par une phase expérimentale.

Le document reprendra la méthodologie utilisée durant ce stage. Dans un premier chapitre les travaux précédents d'asservissement visuel seront étudiés et les concepts seront définis. Dans un deuxième chapitre seront rappelés les travaux réalisés dans le domaine de l'asservissement visuel omnidirectionnel puis l'élaboration de la méthode d'asservissement visuel omnidirectionnelle proposée sera présentée. Par la suite le processus de navigation utilisant cet asservissement visuel sera étudié après un bref état de l'art. Dans chacune des parties les tests réalisés seront décrits et leurs résultats développés.

1

Asservissement visuel - Généralités

L'asservissement visuel est le processus permettant de contrôler un robot en utilisant l'information recueillie par une ou plusieurs caméras, caméras qui peuvent être situées au bout du robot (systèmes de type *Eye-in-Hand*) ou simplement observer ce dernier (systèmes de type *Eye-to-Hand*). Ici les systèmes considérés seront de type *Eye-in-Hand*. Les actions alors effectuées vont alors du simple positionnement à des tâches plus complexes comme la navigation. Cette partie débutera donc par la description de principes d'asservissement visuel pour des tâches de positionnement avant d'étendre le problème à la navigation de robots.

1.1 Tâches de positionnement

Ce paragraphe a pour but d'expliquer le principe d'une tâche de positionnement. Pour positionner un robot il faut agir sur ses degrés de liberté, or les informations recueillies par le robot sont visuelles. Pour effectuer la tâche choisie il faut donc pouvoir relier ces deux concepts. Voici un schéma explicatif de cette boucle fermée perception-action :

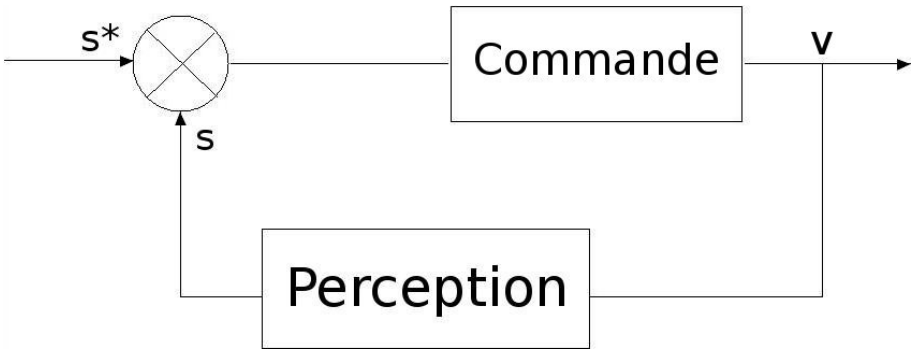


FIGURE 1.1 – Schéma d'asservissement visuel

A chaque instant une pose du robot est déterminée. Notée $\mathbf{r}$, elle représente la translation et la rotation de la caméra. Le but va alors être de minimiser l'erreur entre $\mathbf{r}$ et la pose souhaitée $\mathbf{r}^*$. Pour cela il convient de déterminer une fonction de coût f qui traduit l'erreur entre les deux positions. La tâche de commande peut alors être exprimée comme ceci :

$$\hat{\mathbf{r}} = \underset{\mathbf{r}}{argmin} \ \ f(\mathbf{r}, \mathbf{r}^*) \tag{1.1}$$

où $\hat{\mathbf{r}}$ est la position réellement atteinte par le robot. Il s'agit alors d'un problème de minimisation de la fonction d'erreur f qui, si elle est bien choisie, doit être nulle à son minimum. À chaque pas de la boucle d'asservissement un vecteur de vitesse à appliquer à chaque degré de liberté du robot est calculé ce qui donne une nouvelle position $\mathbf{r}$ qui sera utilisée pour la prochaine itération de la boucle. En se basant sur ce modèle il est alors possible de distinguer deux types d'approches pour définir f : celles qui se basent sur des primitives géométriques extraites des images (et qui nécessitent donc une étape de suivi des-dites primitives) et celles qui se basent sur d'autres types d'informations contenues dans les images.

1.2 Positionnement par asservissement visuel géométrique

La plupart des approches sont dites géométriques. Ces dernières utilisent des primitives géométriques 2D ou 3D calculées à partir d'informations extraites des images acquises pour créer la fonction de coût qui va être minimisée. Ces informations sont alors stockées dans un vecteur $\mathbf{s(r)}$ qu'il va donc falloir chercher à rapprocher du vecteur correspondant à la position souhaitée $\mathbf{s}^*$. La fonction de coût est donc définie comme l'erreur $\mathbf{e} = \mathbf{s}(\mathbf{r}) - \mathbf{s}^*$ entre les deux vecteurs et la tâche s'écrit alors :

$$\hat{\mathbf{r}} = \underset{\mathbf{r}}{argmin} \ \ \|\mathbf{e}(\mathbf{r})\|. \tag{1.2}$$

La position désirée est fixée et ne doit donc pas changer au cours du temps. Il en résulte que la dérivée de $\mathbf{e}$ notée $\dot{\mathbf{e}}$ est égale à la variation temporelle des informations géométriques observées soit $\dot{\mathbf{s}}$. Comme le robot bouge $\dot{\mathbf{s}}$ est fonction du mouvement du robot et des variations créées dans l'image par ce mouvement soit :

$$\dot{\mathbf{s}} = \mathbf{L_s v} \tag{1.3}$$

où $\mathbf{v}$ est la vitesse appliquée à la caméra et $\mathbf{L_s}$ est la matrice d'interaction de la tâche d'asservissement visuel [4]. Cette matrice relie la variation des informations géométriques dans l'image à la vitesse appliquée au robot. À partir de la variation de $\mathbf{s}$ oil est donc possible de déterminer une vitesse $\mathbf{v}$. Mais en pratique une contrainte supplémentaire est ajoutée, en spécifiant une décroissance exponentielle de l'erreur $\mathbf{e}$ soit $\dot{\mathbf{e}} = -\lambda \mathbf{e}$. Ceci va permettre de voir évoluer la vitesse de manière importante lorsque le robot se trouve loin

de la position voulue et de plus en plus lentement à mesure que l'erreur tend vers 0. La nouvelle règle à suivre est alors la suivante :

$$\dot{\mathbf{e}} = -\lambda \mathbf{e} \qquad \text{soit} \qquad \mathbf{L_s v} = -\lambda(\mathbf{s}(\mathbf{r}) - \mathbf{s}^*) \tag{1.4}$$

Ce système est résolu par une approche de type moindres carrés et aboutit à une solution de la forme :

$$\mathbf{v} = -\lambda \mathbf{L_s^+ e} \tag{1.5}$$

avec $\mathbf{L_s^+}$ définie comme la pseudo-inverse de $\mathbf{L_s}$: $\mathbf{L_s^+} = (\mathbf{L}_s^T \mathbf{L}_s)^{-1} \mathbf{L}_s^T$. Cette formule n'étant correcte que si $\mathbf{L_s}$ est de rang plein, la technique nécessite au moins autant d'informations visuelles que de degrés de liberté à commander.

Pour effectuer cet asservissement il est nécessaire d'extraire des informations dans l'image, par exemple suivre un ensemble de points qui sert à la localisation de la position actuelle pour ensuite effectuer la minimisation. Or cette phase demande un suivi ou une mise en correspondance (*matching*) efficace, élimine une grosse partie des informations contenues dans l'image et peut entraîner des erreurs qui vont alors se propager lors de la tâche d'asservissement visuel. Pour éviter ces problèmes de nouvelles approches de type direct ont récemment été proposées.

1.3 Positionnement par asservissement visuel direct

Afin de s'affranchir des processus de traitement et d'extraction de primitives dans l'image et de se servir de toute l'information présente dans ces dernières, il est nécessaire de considérer d'autres mesures visuelles pour déterminer l'erreur entre les positions désirée et actuelle. Dans le cadre de ce rapport ce sont les approches décrites dans [6] et [8] qui seront étudiées, même si d'autres semblent très intéressantes comme par exemple [5].

1.3.1 Asservissement photométrique

L'image elle même peut être considérée comme information sur laquelle baser la loi de commande. Cela donne :

$$\mathbf{s}(\mathbf{r}) = \mathbf{I}(\mathbf{r}) \tag{1.6}$$

soit :

$$\mathbf{e}(\mathbf{r}) = \|\mathbf{I}(\mathbf{r}) - \mathbf{I}^*\| = (\mathbf{I}(\mathbf{r}) - \mathbf{I}^*)^T (\mathbf{I}(\mathbf{r}) - \mathbf{I}^*) \tag{1.7}$$

Dans [6] une fonction de coût reposant sur les intensités des pixels est proposée. Cette fonction a pour hypothèse principale le fait que la luminosité d'un point $\mathbf{x}$, le projeté d'un

point 3D $\mathbf{X}$, est constante au cours du temps soit :

$$I(\mathbf{x} + \mathbf{dx}, t + dt) = I(\mathbf{x}, t) \tag{1.8}$$

où $\mathbf{dx}$ correspond au déplacement effectué pendant le laps de temps dt. Un développement de Taylor d'ordre 1 permet d'exprimer $\dot{I}$ la dérivée de $I(\mathbf{r})$ par rapport au temps :

$$\dot{I} = \mathbf{L}_I \mathbf{v} \qquad \text{avec} \qquad \mathbf{L}_I = -\nabla I^T \mathbf{L}_\mathbf{x}. \tag{1.9}$$

$\mathbf{L}_I(\mathbf{x})$ correspond à la variation de la luminance d'un point $\mathbf{x}$ suivant le mouvement de la caméra. Elle est donnée par la formule suivante :

$$\mathbf{L}_I = -(\nabla I_x \mathbf{L}_x + \nabla I_y \mathbf{L}_y) \tag{1.10}$$

où ∇I_x et ∇I_y sont les composantes du gradient de $\mathbf{I}$ au point $\mathbf{x}$ et $\mathbf{L}_x$ et $\mathbf{L}_y$ sont les matrices d'interaction associées aux coordonnées x et y de $\mathbf{x}$. Ceci n'étant valable que pour un pixel, si l'on considère l'image $\mathbf{I(r)}$ (pour une image de taille m par n) :

$$\mathbf{I}(\mathbf{r}) = (I(\mathbf{x_{00}}) \dots I(\mathbf{x_{mn}})) \tag{1.11}$$

la matrice d'interaction est donnée par :

$$\mathbf{L_I} = \begin{pmatrix} \mathbf{L}_{I(\mathbf{x_{00}})} \\ \vdots \\ \mathbf{L}_{I(\mathbf{x_{mn}})} \end{pmatrix}. \tag{1.12}$$

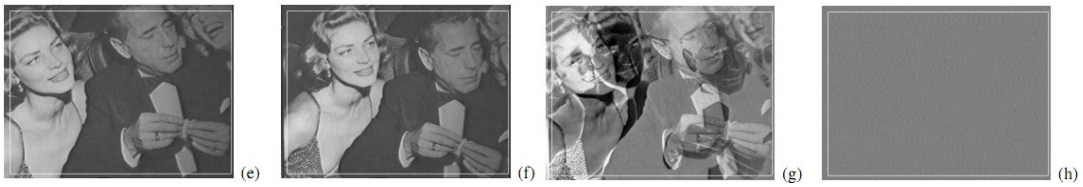


FIGURE 1.2 – Cette Figure est tirée de [6]. Ici (e) représente l'image initiale et (f) l'image désirée (et aussi finale) de l'asservissement visuel. (g) et (h) sont alors les erreurs associées. L'erreur est nulle à la fin, ce qui veut dire que l'asservissement visuel est un succès.

Cette méthode montre la possibilité d'utiliser les valeurs de luminance sur l'ensemble de l'image pour effectuer une tâche de positionnement. Il a été montré en [6] que la méthode était robuste vis à vis de petites occlusions ou de réflections dues à des objets non parfaitement diffus. Cependant l'hypothèse de base de l'algorithme (1.8) fait que la méthode est par définition limitée à des scènes qui ne varient pas trop au cours du temps. Le mouvement de la lumière ou les occlusions importantes n'étant pas prises en compte le modèle n'est donc pas adapté à des scènes en variation perpétuelle comme le sont les scènes extérieures.

1.3.2 Asservissement reposant sur l'information mutuelle

En se basant sur [6], d'autres travaux plus récents ont été effectués dans le domaine de l'asservissement visuel direct. Pour s'affranchir de l'hypothèse de conservation de luminance (1.8), [8] a défini une nouvelle fonction de coût pour l'asservissement visuel. La solution choisie est de se servir de l'information contenue dans une image définie au sens de Shannon et appelée entropie [22]. De ce concept est issue une mesure de similarité appelée information mutuelle, qui traduit la quantité d'information partagée par deux images. Cette mesure possède de nombreuses propriétés intéressantes. S'appuyant sur l'information "pure" partagée par les images, elle est robuste aux variations de lumière et aux occultations, ce qui est très appréciable lors de tâches d'asservissement où un robot en mouvement peut rencontrer des variations des conditions de luminosité du fait de son déplacement ou encore passer derrière certains objets et donc créer des occultations. De plus, cette mesure de similarité s'appuyant sur l'information contenue dans le signal, elle va permettre de mettre en rapport des images présentant différentes représentations de la même information. Par exemple, les images d'un axe routier représenté sous la forme d'une carte pourront être mises en correspondance avec celles prises par satellite. De la même manière, des photos prises de jour pourront être comparées à des vues infrarouges prises de nuit. Toutes ces possibilités sont en pratique très utiles lors de la rencontre(fréquente) de bruit ou de flou sur les images. De manière plus formelle, le calcul de l'information mutuelle appliqué à une image est défini par :

$$\mathrm{MI}(\mathbf{I}(\mathbf{r}), \mathbf{I}^*) = \mathrm{H}(\mathbf{I}(\mathbf{r})) + \mathrm{H}(\mathbf{I}^*) - \mathrm{H}(\mathbf{I}(\mathbf{r}), \mathbf{I}^*) \tag{1.13}$$

où $\mathrm{H}(\mathbf{I})$ est l'entropie de la variable aléatoire $\mathbf{I}$ et $\mathrm{H}(\mathbf{I}, \mathbf{I}^*)$ l'entropie jointe de $\mathbf{I}$ et $\mathbf{I}^*$. $\mathbf{I}(\mathbf{r})$ représente l'image capturée à la position $\mathbf{r}$. Ces entropies sont calculées de la manière suivante :

$$\mathrm{H}(\mathbf{I}) = -\sum_{i=0}^{N_{c_\mathbf{I}}} p_\mathbf{I}(i) \log\left(p_\mathbf{I}(i)\right) \tag{1.14}$$

$$\mathrm{H}(\mathbf{I}, \mathbf{I}^*) = -\sum_{i=0}^{N_{c_\mathbf{I}}} \sum_{j=0}^{N_{c_{\mathbf{I}^*}}} p_{\mathbf{I}\mathbf{I}^*}(i,j) \log\left(p_{\mathbf{I}\mathbf{I}^*}(i,j)\right) \tag{1.15}$$

avec N_c la dynamique de l'image (généralement 255), $p_\mathbf{I}(i) = \Pr(\mathbf{I}(\mathbf{x}) = i)$ la distribution de probabilité de i et $p_{\mathbf{I}\mathbf{I}^*}(i,j) = \Pr(\mathbf{I}(\mathbf{x}) = i \cap \mathbf{I}^*(\mathbf{x}) = j)$ la fonction de distribution de la probabilité jointe. Ces probabilités sont mesurées par un calcul d'histogrammes dans les images qui permettent de donner la proportion d'un niveau de gris donné dans une image et donc sa probabilité d'apparition.

Cependant en pratique le calcul de l'information mutuelle de cette manière crée une fonction au maximum global marqué très localement et donc assez difficilement utilisable

dans le cadre d'une maximisation. Les histogrammes aussi posent problème. Leur calcul prend du temps et ils ne sont pas dérivables. De plus ils peuvent entraîner des minima locaux. Pour surmonter ces problèmes, [8] s'inspire des travaux de [24] et de [19] et adapte la formulation de l'information mutuelle. Dans un premier temps une remise à l'échelle des valeurs est effectuée ce qui a pour effet de réduire le nombre de niveaux de gris concernés dans l'histogramme. De plus, il lisse les histogrammes en utilisant des fonctions B-splines (ϕ) afin que l'image présente le plus de niveaux de gris possibles avant la mise à l'échelle. Ceci a pour effet un histogramme mieux réparti et donc meilleur dans le cadre d'une optimisation. Ceci amène aussi la possibilité de dériver la fonction. La probabilité jointe utilisée pour le calcul de l'entropie est ainsi définie par :

$$p_{\mathbf{II}^*}(i,j,\mathbf{r}) = \frac{1}{N_{\mathbf{x}}} \sum_{\mathbf{x}} \phi\left(i - \overline{\mathbf{I}}(\mathbf{r},\mathbf{x})\right) \phi\left(j - \overline{\mathbf{I}^*}(\mathbf{x})\right) \tag{1.16}$$

où $N_{\mathbf{x}}$ représente le facteur de mise à l'échelle, soit le nombre final de valeurs dans l'histogramme.

La fonction de coût étant définie il faut créer la loi de commande pour la tâche d'asservissement visuel. La ressemblance entre deux images est maximale lorsque leur information mutuelle est maximale. La loi de commande utilisée devra donc être adaptée afin de prendre en compte cela et d'effectuer une maximisation. Comme une fonction est nulle à son maximum la loi de commande doit chercher à annuler la dérivée de l'information mutuelle. La loi proposée est de la forme :

$$\mathbf{v} = -\lambda \hat{\mathbf{L}}_{\mathbf{e}}^{+} \mathbf{e} \tag{1.17}$$

où $\hat{\mathbf{L}}_{\mathbf{e}}^{+}$ est l'estimation de la pseudo-inverse associée à la tâche $\mathbf{e}$. Ici la tâche est identifiée par la matrice d'interaction du gradient de l'information mutuelle notée $\mathbf{L}_{MI}^{T}$ qui doit être nul à convergence et donc (1.17) devient :

$$\mathbf{v} = -\lambda \mathbf{H}_{MI}^{-1} \mathbf{L}_{MI}^{T}. \tag{1.18}$$

où $\mathbf{H_{MI}}$ est la matrice d'interaction associée à $\mathbf{L}_{MI}$ appelée matrice Hessienne de MI. $\mathbf{v}$ et $\mathbf{L}_{MI}$ étant de mêmes dimensions, $\mathbf{H}_{MI}$ est directement inversible. Ces matrices sont définies dans [7] comme :

$$\mathbf{L}_{MI} = \sum_{i,j} \frac{\partial p_{\mathbf{II}^*}}{\partial \mathbf{r}} \left(1 + \log\left(\frac{p_{\mathbf{II}^*}}{p_{\mathbf{I}}}\right)\right) \tag{1.19}$$

$$\begin{aligned} \mathbf{H}_{MI} &= \frac{\partial \mathbf{L}_{MI}}{\partial \mathbf{r}} \\ &= \sum_{i,j} \frac{\partial p_{\mathbf{II}^*}}{\partial \mathbf{r}}^{\top} \frac{\partial p_{\mathbf{II}^*}}{\partial \mathbf{r}} \left(\frac{1}{p_{\mathbf{II}^*}} - \frac{1}{p_{\mathbf{I}}}\right) + \frac{\partial^2 p_{\mathbf{II}^*}}{\partial \mathbf{r}^2} \left(\frac{p_{\mathbf{II}^*}}{p_{\mathbf{I}}}\right). \end{aligned} \tag{1.20}$$

La dérivation de (1.16) conduit à :

$$\frac{\partial p_{\mathbf{II}^*}(i,j,\mathbf{r}))}{\partial \mathbf{r}} = \frac{1}{N_{\mathbf{x}}} \sum_{\mathbf{x}} \frac{\partial \phi}{\partial \mathbf{r}} \big(i - \bar{\mathbf{I}}(\mathbf{r},\mathbf{x})\big)\, \phi\big(j - \overline{\mathbf{I}^*}(\mathbf{x})\big) \tag{1.21}$$

$$\frac{\partial^2 p_{\mathbf{II}^*}(i,j,\mathbf{r}))}{\partial \mathbf{r}^2} = \frac{1}{N_{\mathbf{x}}} \sum_{\mathbf{x}} \frac{\partial^2 \phi}{\partial \mathbf{r}^2} \big(i - \bar{\mathbf{I}}(\mathbf{r},\mathbf{x})\big)\, \phi\big(j - \overline{\mathbf{I}^*}(\mathbf{x})\big) \tag{1.22}$$

avec :

$$\frac{\partial \phi}{\partial \mathbf{r}} = -\frac{\partial \phi}{\partial i} \mathbf{L}_{\bar{\mathbf{I}}} \tag{1.23}$$

$$\frac{\partial^2 \phi}{\partial^2 \mathbf{r}} = \frac{\partial^2 \phi}{\partial^2 i} \mathbf{L}_{\bar{\mathbf{I}}}^{\mathbf{T}} \mathbf{L}_{\bar{\mathbf{I}}} - \frac{\partial \phi}{\partial \mathbf{i}} \mathbf{H}_{\bar{\mathbf{I}}} \tag{1.24}$$

et :

$$\mathbf{L}_{\bar{\mathbf{I}}} = \nabla \bar{\mathbf{I}}\, \mathbf{L}_{\mathbf{x}} \tag{1.25}$$

$$\mathbf{H}_{\bar{\mathbf{I}}} = \mathbf{L}_{\mathbf{x}}{}^{T} \nabla^2 \bar{\mathbf{I}}\, \mathbf{L}_{\mathbf{x}} + \nabla \bar{\mathbf{I}}\, \mathbf{H}_{\mathbf{x}} \tag{1.26}$$

Dans [8], des tests expérimentaux ont confirmé les propriétés de l'information mutuelle. Des changements locaux d'illumination n'empêchent pas la convergence de l'algorithme. Des tests sur des situations d'occultations ou de positionnement par rapport à des scènes non planaires donnent aussi des erreurs résiduelles finales très faibles. Un test sur des images multimodales a été réalisé. Une séquence d'images d'un parcours aérien au dessus d'une carte IGN a été acquise et l'asservissement visuel a été lancé sur des images satellite du même endroit avec comme référence la carte. C'est grâce aux propriétés de l'information mutuelle qui se base sur l'information contenue dans l'image plutôt que sur la luminosité des pixels que le robot a quand même pu suivre le chemin.

Conclusion

Dans ce chapitre, les notions de base sur l'asservissement visuel ont été posées. L'asservissement visuel permet de contrôler un robot grâce aux informations visuelles acquises avec une ou plusieurs caméras. Plusieurs approches existent pour cela. Elles sont réparties en deux catégories. Les approches dites géométriques se basent sur des primitives géométriques extraites des images tels des points ou des droites pour repérer le robot. Les approches directes quant à elles se basent sur l'ensemble de l'information contenue dans l'image. Il n'en existe aucune d'intrinsèquement meilleure que les autres, chacune ayant ses avantages et ses inconvénients et correspondant donc mieux à une situation particulière.

Cependant, toutes ces méthodes se basent sur des caméras *perspective*. Or ces caméras ont un champ de vision restreint, ce qui limite fortement la quantité d'information visuelle. C'est en partant de ce constat que des travaux ont été effectués sur des caméras omnidirectionnelles.

2
Asservissement visuel omnidirectionnel

2.1 État de l'art

Les travaux décrits jusqu'ici se basent sur des images acquises par des caméras *perspective*. L'inconvénient majeur de ce type de caméra est leur champ de vision réduit qui limite la quantité d'information visuelle. Pour illustrer ce problème il convient d'imaginer un robot qui se déplace grâce à un asservissement visuel. Imaginons qu'il détecte un obstacle et doive l'éviter. La modification de trajectoire peut alors entraîner le fait que le robot ne voie plus de rapport entre ce qu'il perçoit et son but. La tâche d'asservissement serait alors un échec. C'est pour palier ce genre de problèmes que des travaux ont été effectués utilisant des caméras omnidirectionnelles. Ces caméras suivent un modèle de projection différent dit *projection centrale* et permettent un champ de vision approchant les 180°. Ces travaux redéfinissent complètement les modèles présentés dans le chapitre précédent afin des les adapter à cette nouvelle projection.

Ce chapitre rappellera donc dans un premier temps des notions sur le modèle de caméra utilisé ainsi que des travaux effectués dans le domaine. Par la suite les méthodes développées durant ce stage seront décrites avant que leurs résultats soient commentés.

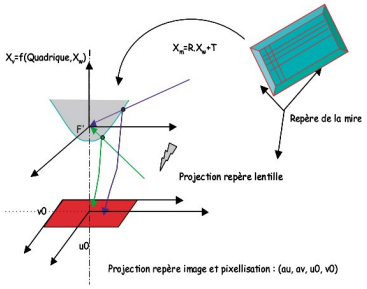
2.1.1 Modélisation

La manière d'acquérir une image pour une caméra omnidirectionnelle est différente de celle utilisée par les caméras *perspective*. Ici la lumière arrivant au capteur est réfléchie par un miroir qui introduit à cette occasion une transformation qui devra être prise en compte lors de toute tâche de traitement d'image.

Suivant le miroir utilisé plusieurs transformations sont possibles. Les différents types de miroirs et les transformations associées peuvent être trouvées dans [17] et [2] décrit le modèle géométrique considéré. La modélisation des paramètres de l'optimisation nécessaire pour l'asservissement doit être adaptée en conséquence. Par exemple, il a été expliqué dans le chapitre 1 que la matrice d'interaction $\mathbf{L_s}$ lie les mouvements du point dans l'image à ceux de la caméra. Ces mouvements sont différents pour chaque projection. Les gradients

aussi peuvent changer. En effet les points étant projetés différemment leurs vrais voisins ne sont pas forcement ceux directement à côté d'eux dans l'image.

(a) Exemple d'image catadioptrique.



(b) Schéma général des transformations subies par la projection de l'objet bleu dans l'image catadioptrique.

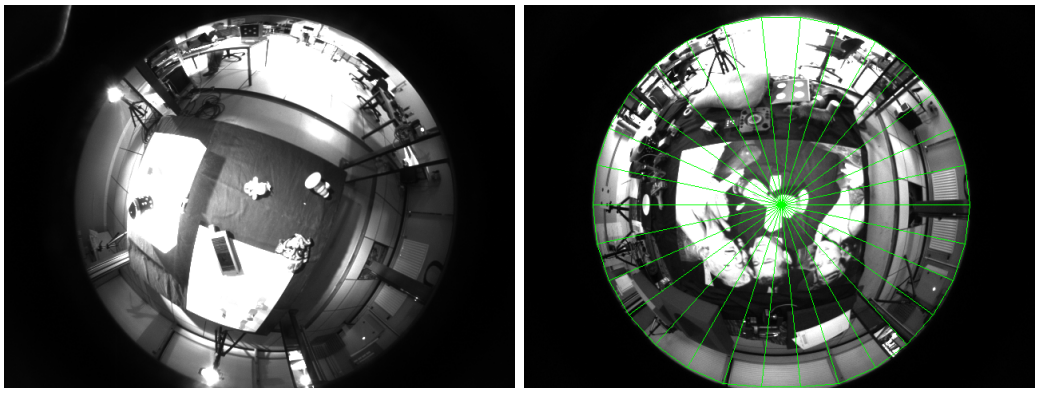
FIGURE 2.1 – Figures tirées de [17].

2.1.2 Asservissement visuel direct

S'appuyant sur les travaux définissant l'asservissement visuel photométrique ([6]), [3] décrit un asservissement visuel direct omnidirectionnel. Le modèle de projection utilisé pour les caméras est ici le modèle de projection centrale proposée par Barreto ([2]), adapté aux caméras *fisheye*. L'article décrit alors le processus d'asservissement visuel direct adapté au modèle considéré. Les différences avec l'asservissement utilisant des caméras *perspective* se situent au niveaux de la représentation des informations visuelles. En effet, un point étant projeté de manière différente il est logique que le résultat de sa projection varie différemment lorsque le robot bouge. Deux modèles de représentation des informations visuelles sont proposés. Le premier, la paramétrisation cartésienne sphérique, projette des points 3D sur la sphère de la caméra en utilisant les équations du modèle tandis que le deuxième modèle se base sur les points projetés dans l'image. Ces deux modèles mènent alors à deux formulations différentes de la matrice d'interaction et à deux algorithmes d'asservissement visuel s'appuyant sur différentes méthodes de calcul des gradients. Les différentes expériences alors réalisées montrent que travailler sur la sphère d'équivalence avec une paramétrisation sphérique cartésienne permet une convergence rapide ainsi qu'un cône de convergence plus grand. Les expériences montrent également que les calculs des gradients sphériques amènent à des convergences bien meilleures que les gradients calculés sur le plan.

2.2 Asservissement visuel omnidirectionnel entropique

Un des objectifs de ce stage est de partir des travaux existants avec des caméras *perspective* ([8]) et de les adapter aux caméras omnidirectionnelles. Pour cela il faut redéfinir toutes les parties de l'algorithme liées à la projection d'un point par la caméra. En pratique, trois étapes sont très impactées. La première est la zone de l'image suivie. Comme les images ci-dessous le montrent, l'information est dans un disque entouré de noir. Il faut donc pouvoir délimiter la zone d'intérêt. La deuxième étape est le calcul des gradients. Du fait de la réflexion de la lumière sur une surface non plane la résolution n'est pas la même en tous points de l'image et les gradients doivent donc prendre ceci en compte. Enfin la dernière étape est celle de la matrice d'interaction $\mathbf{L_x}$. Cette matrice lie le mouvement d'un point dans l'image au mouvement de la caméra et est donc différente suivant le modèle de projection utilisé.



(a) Exemple de prise de vue de la caméra *fisheye*.

(b) Initialisation de la zone d'intérêt.

FIGURE 2.2 – Images acquises par la caméra.

Pour ce qui est de la détermination de la zone d'intérêt la méthode choisie consiste à partir du centre optique de la caméra puis à échantillonner des points sur la circonférence du disque afin de créer un ensemble de triangles qui définissent la zone considérée.

Pour ce qui est des deux autres points il faut déterminer les modèles de représentation sur lesquels travailler avant de pouvoir les détailler. Le modèle utilisé est le modèle de projection unifié proposé par Barreto[2] pour une famille de caméras incluant les caméras catadioptriques (il a été montré dans [27] que ce modèle était adapté aux caméras *fisheye*). Ce modèle décrit la projection d'un point 3D sur le plan image par une double projection. Dans un premier temps le point est projeté en 3D sur une sphère unitaire puis ce dernier est projeté sur le plan image.

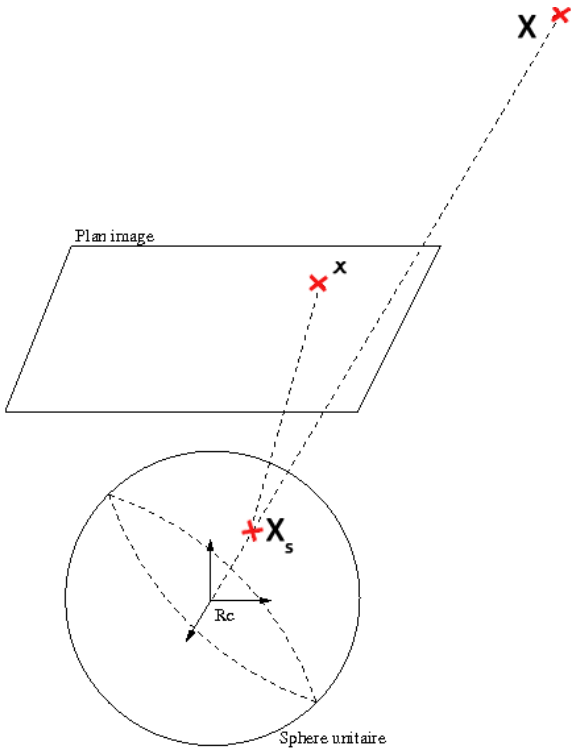


FIGURE 2.3 – Modèle de projection unifié, Rc représente le repère de la caméra.

Plus formellement, le point $\mathbf{X}$ $(X,Y,Z)^T$ est projeté dans le plan image suivant :

$$\mathbf{x} = pr_\gamma(\mathbf{X}) \quad \text{avec} \quad \begin{cases} x = \frac{X}{Z+\xi\rho} \\ y = \frac{Y}{Z+\xi\rho} \end{cases} \tag{2.1}$$

avec ρ défini par $\sqrt{X^2+Y^2+Z^2}$. À partir de cette projection le point dans l'image résulte d'une simple conversion mètre-pixel :

$$\mathbf{u} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} p_x & 0 & u_0 \\ 0 & p_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathbf{K}\mathbf{x} \tag{2.2}$$

Pour éviter la perte de précision due à une seconde projection il est également possible de s'intéresser au point issu de la première projection sur une sphère d'équivalence :

$$\mathbf{X}_\mathcal{S} = pr_\mathcal{S}(\mathbf{X}) \quad \text{with} \quad \begin{cases} X_\mathcal{S} = \frac{X}{\rho} \\ Y_\mathcal{S} = \frac{Y}{\rho} \\ Z_\mathcal{S} = \frac{Z}{\rho} \end{cases} \tag{2.3}$$

Ce point peut également être obtenu grâce à la projection inverse du point dans le plan image, ce qui donne :

$$\mathbf{X}_{\mathcal{S}} = pr_{\gamma}^{-1}(\mathbf{x}) = \begin{pmatrix} \frac{\xi+\sqrt{1+(1-\xi^2)(x^2+y^2)}}{x^2+y^2+1}x \\ \frac{\xi+\sqrt{1+(1-\xi^2)(x^2+y^2)}}{x^2+y^2+1}y \\ \frac{\xi+\sqrt{1+(1-\xi^2)(x^2+y^2)}}{x^2+y^2+1} - \xi \end{pmatrix} \tag{2.4}$$

Deux méthodes se distinguent alors. Les calculs peuvent être effectués dans le plan image et donc sur le point $\mathbf{x}$ ou plutôt sur la sphère, soit sur $\mathbf{X_s}$. Ne pouvant affirmer la supériorité d'un modèle sur l'autre, l'approche choisie a donc été de considérer les deux méthodes puis de comparer leurs résultats expérimentaux.

2.2.1 Méthode avec gradients planaires

La première méthode a donc été de considérer le plan image. Les gradients sont alors calculés via un filtre dérivatreur défini par :

$$\frac{1}{8418}\,[\,-112 \quad -913 \quad -2047 \quad 0 \quad 2047 \quad 913 \quad 112].$$

Ce filtre est appliqué sur les voisins directs dans l'image et les différences de résolution dans l'image dues au modèle de projection ne sont donc pas prises en compte. Comme il a été montré dans le premier chapitre, la tâche d'asservissement visuel est directement liée à la matrice d'interaction $\mathbf{L_x}$ comme l'atteste l'équation (2.5) :

$$\begin{aligned} \mathbf{L}_{\bar{\mathbf{I}}} &= \nabla\bar{\mathbf{I}}\,\mathbf{L_x} \\ \mathbf{H}_{\bar{\mathbf{I}}} &= \mathbf{L_x}^T\nabla^2\bar{\mathbf{I}}\,\mathbf{L_x} + \nabla\bar{\mathbf{I}}\,\mathbf{H_x} \end{aligned}$$

Étant donné qu'elle lie les mouvement d'un point dans l'image aux déplacements du robot, il faut que celle-ci soit redéfinie pour être en adéquation avec le nouveau modèle de projection. En théorie $\mathbf{H_x}$ suit la même logique mais comme pour la tâche d'asservissement la matrice hessienne est estimée à convergence ([9]), $\mathbf{H}_{\bar{\mathbf{I}}}$ peut être négligé ce qui rend inutile le recalcul de $\mathbf{H_x}$. Pour le calcul de $\mathbf{L_x}$, la dérivation de (2.1) peut être trouvée dans [13] et aboutit à :

$$\mathbf{L_x} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \end{pmatrix} \tag{2.5}$$

avec :

$$\begin{aligned}
\mathbf{A} &= \begin{pmatrix} -\frac{1+x^2(1-\xi(\gamma+\xi))+y^2}{\rho(\gamma+\xi)} & \frac{\xi xy}{\rho} & \frac{\gamma x}{\rho} \\ \frac{\xi xy}{\rho} & -\frac{1+y^2(1-\xi(\gamma+\xi))+x^2}{\rho(\gamma+\xi)} & \frac{\gamma y}{\rho} \end{pmatrix} \\
\mathbf{B} &= \begin{pmatrix} xy & -\frac{(1+x^2)\gamma-\xi y^2}{\gamma+\xi} & y \\ \frac{(1+y^2)\gamma-\xi x^2}{\gamma+\xi} & -xy & -x \end{pmatrix} \\
\gamma &= \sqrt{1+(1-\xi^2)(x^2+y^2)}.
\end{aligned}$$

2.2.2 Méthode avec gradients sphériques

La seconde méthode a été de considérer le point $\mathbf{X}_{\mathcal{S}}$. Les calculs ne se font donc plus dans le plan de l'image mais sur une sphère d'équivalence autour du point 3D. La matrice d'interaction associée à cette représentation a été définie par [14] comme :

$$\mathbf{L_x} = \frac{\partial \mathbf{X}_S}{\partial \mathbf{X}} \frac{\partial \mathbf{X}}{\partial \mathbf{r}} = \begin{pmatrix} \frac{1}{\rho}(\mathbf{X}_{\mathcal{S}}\mathbf{X}_{\mathcal{S}}^T - \mathbf{I}_3) & [\mathbf{X}_{\mathcal{S}}]_\times \end{pmatrix}. \tag{2.6}$$

Avec cette approche ce qui change vraiment est le calcul des gradients. Ceux-ci doivent être calculés sur la sphère et il faut donc déterminer les voisins d'un point de cette dernière. Pour ces calculs c'est une méthode proposée par [3] qui a été choisie. Un échantillonnage 3D (un en X, un en Y et un en Z) est effectué autour de chaque point de la zone d'intérêt projeté sur la sphère. Ces voisinages sont alors reprojetés dans l'image et leurs coordonnées sont stockées. Ce processus n'est donc à effectuer qu'une fois au lancement de l'application. Lors de l'application du filtre dérivateur sur chacune des dimensions ce sont donc ces coordonnées qui seront utilisées comme voisins du point et non ses voisins dans l'image.

2.3 Seconde approche : l'information mutuelle normalisée

Durant l'élaboration de ces méthodes, une autre approche a été considérée. Cela a été fait car l'information mutuelle comme définie en (1.13) n'est pas bornée vers le haut. Cela empêche d'évaluer si deux images sont proches par le biais de cette mesure. En effet si les valeurs maximales changent d'une image à l'autre il n'est pas possible de déterminer un seuil à partir duquel deux images sont considérées comme proches. Pour remédier à cela une solution possible était de partir de la définition d'une information mutuelle normalisée exprimée dans [23] :

$$\mathbf{NMI}(\mathbf{I}, \mathbf{I}^*) = \frac{\mathrm{H}(\mathbf{I}) + \mathrm{H}(\mathbf{I}^*)}{\mathrm{H}(\mathbf{I}, \mathbf{I}^*)}. \tag{2.7}$$

Cette mesure a pour propriétés d'être plus robuste lors de situations de recouvrement que l'information mutuelle classique (voir [23]) et d'être comprise entre 1 et 2. En effet lorsque

X et Y sont indépendants alors $H(\mathbf{I},\mathbf{I}^*) = \mathbf{H}(\mathbf{I}) + \mathbf{H}(\mathbf{I}^*)$ donc $\mathbf{NMI}(\mathbf{I},\mathbf{I}^*)$=1. Pour la borne supérieure, par définition :

$$H(\mathbf{I},\mathbf{I}^*) \geq \mathbf{max}\left[\mathbf{H}(\mathbf{I}),\mathbf{H}(\mathbf{I}^*)\right] \text{ et } \mathbf{H}(\mathbf{I},\mathbf{I}^*) > \mathbf{0} \tag{2.8}$$

soit :

$$\frac{1}{H(\mathbf{I},\mathbf{I}^*)} \leq \frac{1}{max\left[H(\mathbf{I}),\mathbf{H}(\mathbf{I}^*)\right]} \tag{2.9}$$

d'où, en posant $\alpha = max\left[H(\mathbf{I}),\mathbf{H}(\mathbf{I}^*)\right]$:

$$\mathbf{NMI} \leq \frac{\alpha(1+\beta)}{\alpha} \text{ avec } \beta \leq 1 \tag{2.10}$$

$$\text{soit } \mathbf{NMI} \leq 1+\beta \tag{2.11}$$

Dans le meilleurs des cas, $\mathbf{I}$ et $\mathbf{I}^*$ ont la même fonction de répartition donc β=1 ce qui entraîne $\mathbf{NMI}$=2. Cette nouvelle définition de l'information mutuelle conduit alors à la reformulation de $\mathbf{L}_{MI}$ et $\mathbf{H}_{MI}$ (1.19 et 1.20). Pour les déterminer il faut partir de la définition de $\mathbf{NMI}$(2.7). En notant son numérateur u et son dénominateur v cela donne :

$$u(\mathbf{r}) = H(\mathbf{I}(\mathbf{r})) + \mathbf{H}(\mathbf{I}^*) \tag{2.12}$$

$$= -\sum_i p_{\mathbf{I}}(i)\log\left(p_{\mathbf{I}}(i)\right) - \sum_j p_{\mathbf{I}^*}(j)\log\left(p_{\mathbf{I}^*}(j)\right) \tag{2.13}$$

$$= \sum_{i,j} -p_{\mathbf{II}^*}(i,j)\log\left(p_{\mathbf{I}}(i)\right) - p_{\mathbf{II}^*}(i,j)\log\left(p_{\mathbf{I}^*}(j)\right) \tag{2.14}$$

$$= -\sum_{i,j} p_{\mathbf{II}^*}(i,j)\log\left(p_{\mathbf{I}}(i)p_{\mathbf{I}^*}(j)\right) \tag{2.15}$$

$$v(\mathbf{r}) = H(\mathbf{I}(\mathbf{r}),\mathbf{I}^*) \tag{2.16}$$

$$= \sum_{i,j} -p_{\mathbf{II}^*}(i,j)\log\left(p_{\mathbf{II}^*}(i,j)\right) \tag{2.17}$$

Pour exprimer $\mathbf{L}_{MI}$ et $\mathbf{H}_{MI}$ leurs dérivées première et seconde seront nécessaires. Comme le template (ici $\mathbf{I}^*$) ne varie pas durant la tâche d'asservissement, les $p_{\mathbf{I}^*}$ sont constantes.

Les calculs donnent :

$$\frac{\partial u(\mathbf{r})}{\partial \mathbf{r}} = -\sum_{i,j} \frac{\partial p_{\mathbf{II}^*}(i,j)}{\partial \mathbf{r}} \log(p_{\mathbf{I}}(i) p_{\mathbf{I}^*}(j)) \quad (2.18)$$

$$\frac{\partial v(\mathbf{r})}{\partial \mathbf{r}} = -\sum_{i,j} \frac{\partial p_{\mathbf{II}^*}(i,j)}{\partial \mathbf{r}} (1 + \log(p_{\mathbf{II}^*}(i,j))) \quad (2.19)$$

$$\frac{\partial^2 u(\mathbf{r})}{\partial \mathbf{r}^2} = -\sum_{i,j} \frac{\partial^2 p_{\mathbf{II}^*}(i,j)}{\partial \mathbf{r}^2} \log(p_{\mathbf{I}}(i) p_{\mathbf{I}^*}(j)) + \frac{1}{p_{\mathbf{II}^*}(i,j)} . \frac{\partial p_{\mathbf{II}^*}(i,j)^2}{\partial \mathbf{r}} \quad (2.20)$$

$$\frac{\partial^2 v(\mathbf{r})}{\partial \mathbf{r}^2} = -\sum_{i,j} \frac{\partial^2 p_{\mathbf{II}^*}(i,j)}{\partial \mathbf{r}^2} (1 + \log(p_{\mathbf{II}^*}(i,j))) + \frac{1}{p_{\mathbf{II}^*}(i,j)} . \frac{\partial p_{\mathbf{II}^*}(i,j)^2}{\partial \mathbf{r}} \quad (2.21)$$

(2.22)

Une fois ces résultats posés $\mathbf{L}_{MI}$ devient :

$$\mathbf{L}_{MI}(\mathbf{r}) = \frac{\frac{\partial u(\mathbf{r})}{\partial \mathbf{r}} . v(\mathbf{r}) - \frac{\partial v(\mathbf{r})}{\partial \mathbf{r}} . u(\mathbf{r})}{v(\mathbf{r})^2} . \quad (2.23)$$

Pour simplifier l'écriture de $\mathbf{H}_{MI}$ le numérateur et le dénominateur de $\mathbf{L}_{MI}$ ainsi que leurs dérivées sont exprimées comme :

$$\alpha(\mathbf{r}) = \frac{\partial u(\mathbf{r})}{\partial \mathbf{r}} . v(\mathbf{r}) - u(\mathbf{r}) . \frac{\partial v(\mathbf{r})}{\partial \mathbf{r}} \quad (2.24)$$

$$\frac{\partial \alpha(\mathbf{r})}{\partial \mathbf{r}} = \frac{\partial^2 u(\mathbf{r})}{\partial \mathbf{r}^2} . v(\mathbf{r}) - u(\mathbf{r}) . \frac{\partial^2 v(\mathbf{r})}{\partial \mathbf{r}^2} \quad (2.25)$$

$$\beta(\mathbf{r}) = v(\mathbf{r})^2 \quad (2.26)$$

$$\frac{\partial \beta(\mathbf{r})}{\partial \mathbf{r}} = 2 * \frac{\partial v(\mathbf{r})}{\partial \mathbf{r}} . v(\mathbf{r}) \quad (2.27)$$

(2.28)

Ce qui donne :

$$\mathbf{H}_{MI}(\mathbf{r}) = \frac{\frac{\partial \alpha(\mathbf{r})}{\partial \mathbf{r}} . \beta(\mathbf{r}) - \alpha(\mathbf{r}) . \frac{\partial \beta(\mathbf{r})}{\partial \mathbf{r}}}{\beta(\mathbf{r})^2} . \quad (2.29)$$

Une fois les calculs effectués, cette méthode a été testée. Cependant les tests ne donnent pas de résultats meilleurs que dans le cas classique. Pour ne pas perdre trop de temps dans le cadre du stage il a alors été décidé de continuer avec l'information mutuelle simple et de revenir sur cette approche en fin de stage pour l'améliorer.

2.4 Résultats

Deux versions de l'asservissement ont donc été développées : Une utilisant les gradients normaux que l'on nommera IPVS (Image Plane Visual Servoing) et l'autre utilisant les gradients sphériques CSVS (Cartesian Spherical Visual Servoing). Pour valider les différents algorithmes, des tests ont été effectués sur un robot cartésien à six degrés de liberté (trois translations, trois rotations).

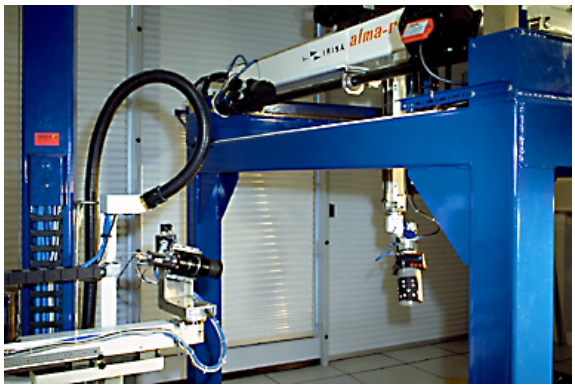
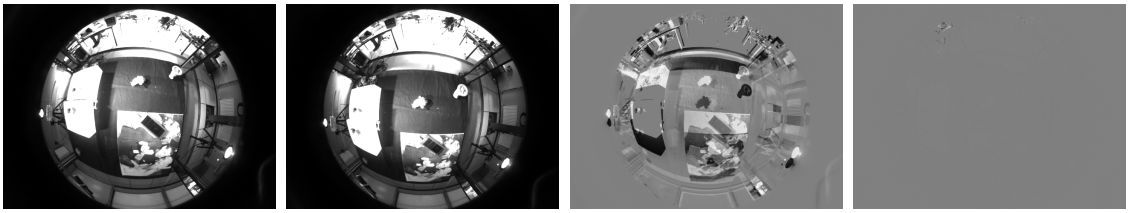


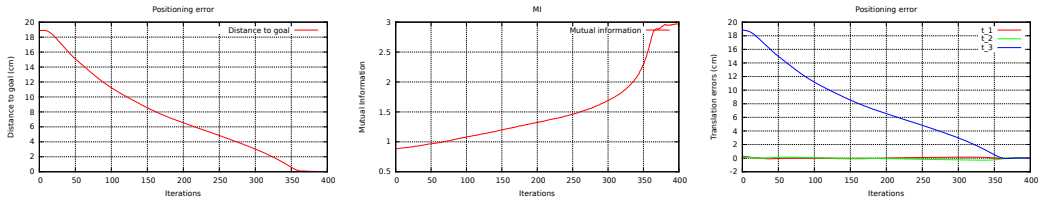
FIGURE 2.4 – Robot cartésien à six degrés de liberté.

Le protocole de test est simple. Le robot capture une image à une position donnée puis il est déplacé à une autre position depuis laquelle l'asservissement est lancé. Le but est alors qu'il retourne à la position désirée. Si ce dernier suit parfaitement le modèle, les erreurs doivent alors décroître de manière exponentielle. Pour déterminer l'efficacité de la méthode il a été décidé d'enregistrer durant le mouvement du robot les erreurs en translation et en rotation entre les deux positions (désirée et réelle) ainsi que la distance topologique entre ces deux positions et la valeur d'information mutuelle entre l'image courante et l'image désirée. Comme il a été dit, le but est que les erreurs décroissent en se rapprochant d'une exponentielle décroissante. Quant à l'information mutuelle le but est de la maximiser. Enfin la position du robot dans l'espace est enregistrée afin de pouvoir interpréter sa trajectoire.
Lors de ces tests le facteur de convergence utilisé (le facteur λ de l'équation (1.18)) suivait une fonction sigmoïde paramétrée par la distance entre les positions désirée et actuelle. Ainsi, plus le robot était loin du but et plus il avançait rapidement. Le réglage de ce gain est alors très important pour faciliter la convergence mais en même temps éviter de possibles oscillations lorsque l'on se rapproche du but dues à un trop fort gain.
Les résultats exposés dans cette section correspondent à une translation du robot. C'est pour cela que seules les erreurs sur les degrés de liberté en translation seront exposées.



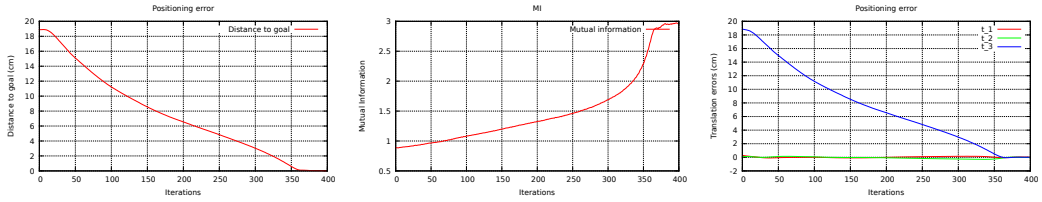
Position de départ *Position désirée* *Erreur initiale* *Erreur finale*

FIGURE 2.5 – Positions de départ et d'arrivée.



Distance au but *Information mutuelle* *Erreurs en translation*

FIGURE 2.6 – Courbes de suivi IPVS.

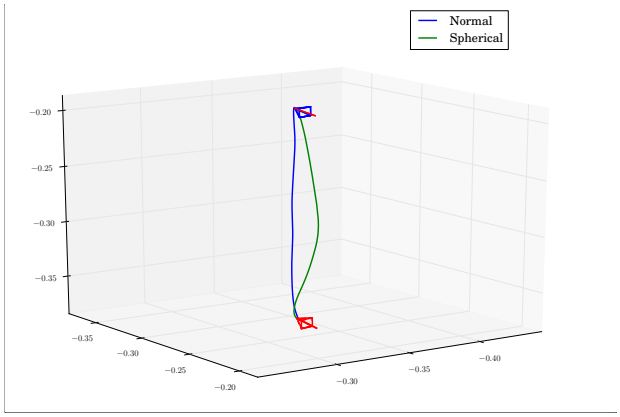


Distance au but *Information mutuelle* *Erreurs en translation*

FIGURE 2.7 – Courbes de suivi CSVS.

Les tests sont alors concluants, la plupart des positions testées permettant d'atteindre le but avec une précision de l'ordre du millimètre. Le premier constat est les deux méthodes fonctionnent et permettent un asservissement visuel depuis des positions qui s'éloignent de manière significative de la position désirée. Les courbes d'erreurs montrent que les décroissances ne sont pas exponentielles, mais ayant altéré la loi de commande par le choix d'un gain adaptable ceci n'est pas très étonnant. Ensuite il ressort des différents tests qu'aucune des deux méthodes ne semble mieux que l'autre en tous points. Elles semblent posséder des domaines de convergence différents, ce qui fait que l'une sera plus efficace que l'autre à certaines positions et inversement. Et ce autant sur le plan des décroissances d'erreurs que des trajectoires.

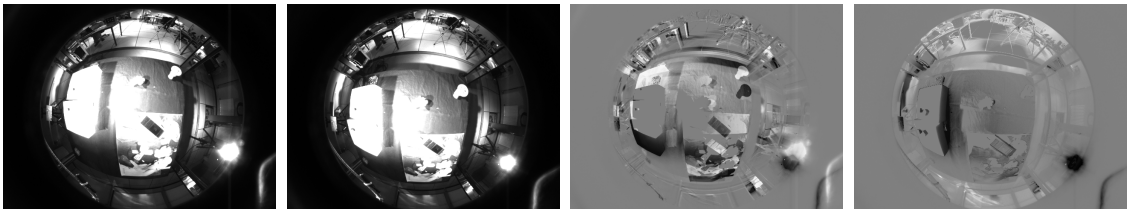
Cependant, il existe un point sur lequel le CSVS semble plus efficace : la robustesse



Trajectoires

FIGURE 2.8 – Comparaison IPVS-CSVS.

aux variations de luminosité et aux occlusions. En effet, comme l'asservissement se base sur l'information mutuelle les méthodes sont sensées être résistantes aux changements de luminosité ainsi qu'aux occlusions. Pour vérifier cela, deux tests ont été effectués. Le premier consite, après avoir enregistré les caractéristiques d'un asservissement visuel depuis une position donnée, à relancer la même expérience en changeant les conditions de luminosité au cours de la tâche. Bien que l'optimisation soit moins efficace que précédemment (la fonction de coût étant rendue plus chaotique), le robot arrive toujours à destination.

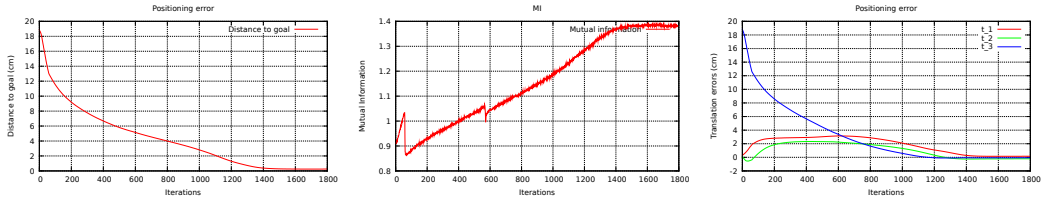


Position de départ *Position désirée* *Erreur initiale* *Erreur finale*

FIGURE 2.9 – Positions de départ et d'arrivée.

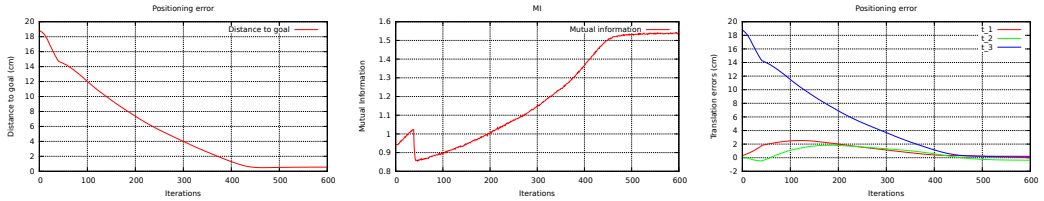
En observant la courbe de l'évolution de l'information mutuelle, des diminutions brutales aparaîssent. Ces dernières sont dues aux changements brutaux de luminosité. L'image d'erreur finale montre elle aussi les changements de luminosité. La courbe de trajectoires montre qu'avec la méthode CSVS le robot a effectué le même déplacement avec et sans les variations. Pour la méthode IPVS en revenche la trajectoire est modifiée et le robot dépassé la position souhaitée avant d'y revenir.

Le deuxième test effectué est alors celui de la résistance aux occultations. Pour tester



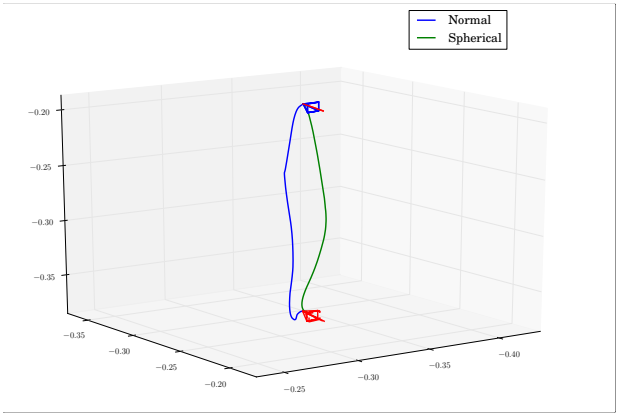
Distance au but *Information mutuelle* *Erreurs en translation*

FIGURE 2.10 – Courbes de suivi IPVS.



Distance au but *Information mutuelle* *Erreurs en translation*

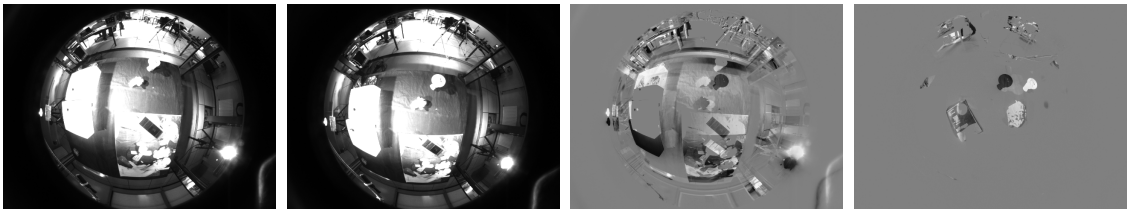
FIGURE 2.11 – Courbes de suivi CSVS.



Trajectoires

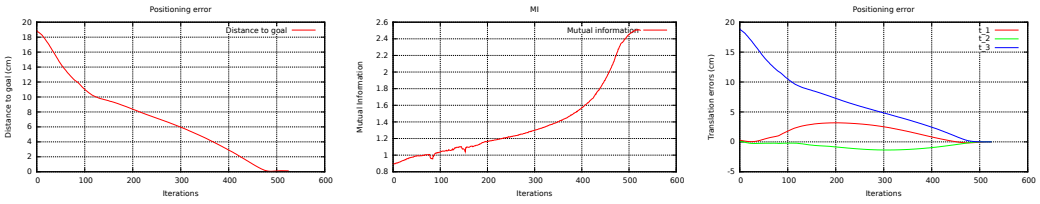
FIGURE 2.12 – Comparaison IPVS-CSVS.

cela des objets présents dans le champ de vision de la caméra ont été déplacés pour analyser le comportement du robot. L'asservissement se passe ici aussi normalement, mais comme pour l'expérience précédente la trajectoire IPVS est plus impactée que la trajectoire CSVS.



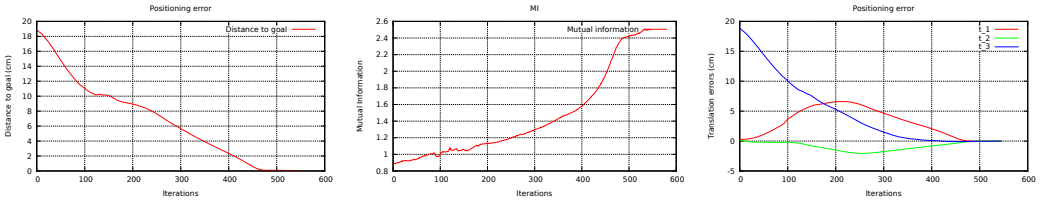
Position de départ *Position désirée* *Erreur initiale* *Erreur finale*

FIGURE 2.13 – Positions de départ et d'arrivée.



Distance au but *Information mutuelle* *Erreurs en translation*

FIGURE 2.14 – Courbes de suivi IPVS.



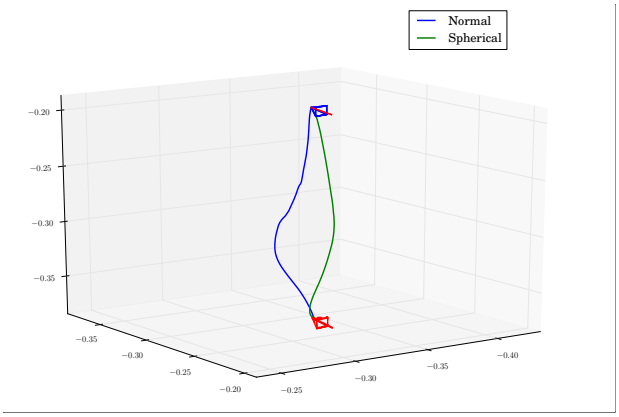
Distance au but *Information mutuelle* *Erreurs en translation*

FIGURE 2.15 – Courbes de suivi CSVS.

2.5 Perspectives

Les deux méthodes développées permettent donc de réaliser un asservissement visuel omnidirectionnel se basant sur l'information mutuelle des images. Les résultats montrent qu'aucune n'est intrinsèquement meilleure que l'autre mais différents tests ont montré que la méthode de CSVS était plus robuste aux changements de conditions de la scène observée.

Mais aucune des deux méthodes n'est parfaite aussi bien du point de vue des décroissances des erreurs que des trajectoires. Plusieurs pistes sont envisageables pour améliorer l'asservissement. Tout d'abord il serait intéressant de considérer une troisième représentation définie comme sphérique pure par [3]. Ensuite c'est principalement sur la méthode d'opti-



Trajectoires

FIGURE 2.16 – Comparaison IPVS-CSVS.

misation que des améliorations paraissent possibles. Premièrement l'information mutuelle ne possède pas de borne haute. Il est donc difficile d'évaluer par cette mesure si deux images sont proches. Une amélioration serait donc de considérer une information mutuelle normalisée ou une autre mesure l'utilisant et qui serait bornée. Cette étude n'ayant pas donné de résultats exploitables pour le moment elle constitue un axe d'amélioration futur. Ensuite, il a été dit dans le chapitre 1 que la mise à l'échelle de l'histogramme lissait la courbe et permettait donc une convergence facilitée. Cependant ce lissage de la courbe lui fait perdre en précision et il serait donc intéressant d'étudier un moyen de raffiner l'histogramme à mesure que l'optimisation se rapproche du but afin de garder la convergence simplifiée mais d'augmenter la précision de positionnement.

Conclusion

Ce chapitre a donc proposé deux nouvelles méthodes d'asservissement visuel se basant sur l'information mutuelle. Il a été montré qu'elles conduisaient à une erreur de positionnement de l'ordre du millimètre et qu'elles étaient robustes aux variations de luminosité et aux occlusions. Leurs résultats ont été comparés et il a été déterminé que la méthode utilisant la représentation du point sur la sphère d'équivalence du modèle de projection considéré était plus robuste. Enfin, les améliorations possibles de ces méthodes ont été débattues. Ces méthodes permettent donc un asservissement visuel. Pour aller plus loin, il est intéressant de les utiliser dans le cadre d'une navigation de robot autonome. En effet leur robustesse pourrait alors se montrer très utile. Le chapitre suivant présentera donc un bref état de l'art des techniques de navigation existantes se basant sur l'asservissement visuel avant que soient décrite l'approche employée.

3 Navigation

3.1 État de l'art

Les parties précédentes ont présenté différentes façons de positionner un robot en s'aidant d'images. Mais ces approches sont locales, et les minimisations effectuées partent du principe que la configuration initiale n'est pas "trop" éloignée de la configuration désirée. En pratique cela revient à dire que la position de départ est assez proche de la position souhaitée. La navigation pose un problème plus général et il est aisé de comprendre pourquoi appliquer une simple tâche de positionnement à une grande trajectoire n'est pas adapté. Si, par exemple, le robot ne peut mesurer d'erreur entre le départ et l'arrivée car les images diffèrent trop la tâche va échouer. Pour effectuer une navigation il faut se ramener à la notion de trajectoire. La trajectoire d'un point étant définie comme l'ensemble des positions successives occupées par ce point au cours du temps, le parallèle avec les tâches de positionnement devient alors évident : suivre une trajectoire c'est effectuer une série de tâches de positionnement. Bien sûr ce n'est pas aussi simple, beaucoup de problèmes se posent comme celui de la détermination de la position désirée pour l'asservissement au cours du temps.

3.1.1 Navigation par asservissement visuel géométrique en 3D

De nombreux articles ont été publiés proposant des méthodes de navigation s'appuyant sur des informations extraites des images.
Dans [20], la navigation se compose de trois étapes. Dans un premier temps le robot est déplacé manuellement le long de la trajectoire et acquiert des images. A partir de cette séquence d'images un algorithme de *structure from motion* [1] est appliqué pour effectuer une reconstruction 3D de l'environnement. C'est en se repérant dans le monde 3D ainsi reconstruit que le robot déterminera son déplacement lors de la tâche de navigation. La navigation est alors vraiment une navigation 3D qui ne se base plus sur les images. La première des trois étapes consiste à balader le robot le long d'une trajectoire et à acquérir des images. Après cette phase il faut construire une représentation 3D du monde traversé.

Pour cela il faut dans un premier temps choisir des images parmi la séquence acquise pour reconstruire. En effet appliquer un algorithme de stéréovision sur la séquence dans son intégralité impliquerait deux choses : la tentative de reconstruction à partir d'images très proches entraînant des erreurs ainsi qu'un temps de calcul très important. Le problème du trop grand nombre d'images se pose également lors de la navigation (en effet si il y a trop d'images choisir la prochaine sur notre chemin devient assez complexe) ce qui fait de la création de la séquence d'images clés un problème central de la navigation. Ces images doivent être à la fois espacées pour simplifier le choix lors de la navigation et augmenter l'efficacité de la reconstruction mais elles doivent également assez être proches pour que le passage de l'une à l'autre se fasse correctement et que les informations extraites soient possible à suivre sur les images successives. Dans [20] ces images sont sélectionnées afin de garder un certain nombre de points de Harris[15] communs entre plusieurs images successives. Le monde est alors reconstruit grâce à la géométrie épipolaire. Des points SIFT[16] sont mis en correspondance dans des images clés. À partir de ce matching la position en 3D du point est définie grâce à des équations de stéréovision qui permettent de déterminer la profondeur du point par triangulation. Une fois le monde reconstruit la phase de navigation peut alors commencer. À partir de la position initiale la navigation se fait dans le monde 3D dans lequel le robot est repéré par un calcul de pose. À chaque nouvelle image une nouvelle pose est calculée par mise en correspondance de points grâce à un algorithme de type RANSAC [11]. Cette étape de mise en correspondance est effectuée sur les points extraits de l'image et ceux de l'image clé courante en se basant sur la pose précédente et un asservissement visuel 3D est alors effectué pour déplacer le robot. Cette méthode est choisie pour minimiser le temps de calcul mais elle peut entraîner des dérives en accumulant des erreurs sur les poses successives et c'est pour cela qu'une phase de *Bundle Adjustment* est également effectuée pour compenser cette erreur.

FIGURE 3.1 – Figure tirée de [20]. A droite les points détectés dans l'image courante sont mis en correspondance avec ceux de l'image référence à gauche afin de calculer la pose de la caméra dans le monde 3D.

Les résultats de ces travaux montrent une bonne précision de la navigation, comparable à la précision d'une navigation via coordonnées GPS dans un environnement ouvert. Ces dernières étant faussées en ville lors d'interférences avec des immeubles par exemple ces deux techniques peuvent être complémentaires suivant les régions de navigation. Cette navigation pose le problème de la connaissance du monde 3D et de sa reconstruction qui déterminent la précision de la trajectoire suivie mais demandent beaucoup de traitements d'images préalables à la navigation.

3.1.2 Utilisation d'un chemin visuel

Dans [21] une autre approche est proposée dans laquelle le monde n'est plus reconstruit en trois dimensions. Ce dernier est représenté comme un environnement hybride topologique-géométrique formé à partir des images clés et stocké dans un graphe dont les sommets sont représentés par des images clés. Ce graphe constitue alors un chemin visuel que le robot devra suivre.

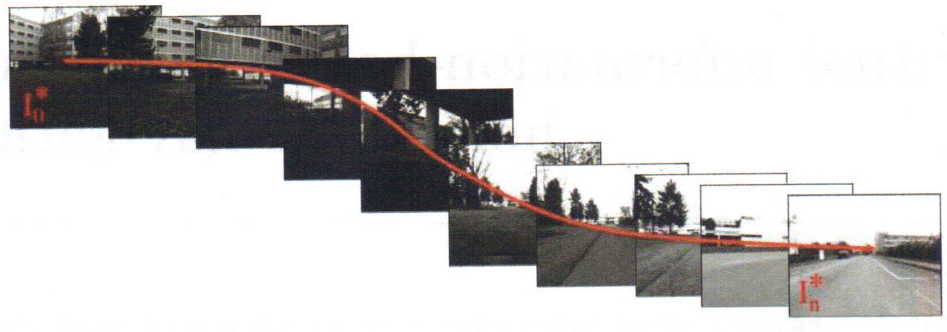


FIGURE 3.2 – Figure tirée de [10]. Illustration de la notion de chemin visuel défini par une suite d'images clés.

L'asservissement visuel est alors fait en deux dimensions et non trois comme dans [20], ce qui permet d'éliminer la localisation 3D et les erreurs qu'elle peut produire. Pour la détermination des images clés deux algorithmes sont ici proposés, un basé sur de la mise en correspondance (*matching*) et un autre basé sur du suivi (*tracking*). Différents tests indiquent que l'algorithme par *matching* produit plus d'images clés que l'algorithme par *tracking*. Les auteurs concluent qu'il vaut mieux utiliser le tracking pour les scènes qui ne présentent pas de difficultés, le matching étant à réserver à des scènes où il faut plus d'images pour des raisons diverses comme par exemple lors d'une occlusion totale par un objet en mouvement. Lors de la navigation, pour déterminer les images clés entre lesquels le robot se trouve, il est proposé de les définir comme les deux images ayant le plus de points d'intérêt en commun avec l'image considérée. Une fois l'image référence déterminée

il s'agit alors d'un asservissement visuel 2D à effectuer sur les points d'intérêt extraits des images et mis en correspondance par une technique de *wide-baseline matching* se basant sur des points de type SIFT[16]. La loi de commande utilisée pour déplacer le robot est de la forme de (5), utilisant les points SIFT précédemment mis en correspondance qui sont suivis d'une itération sur l'autre. Ces points permettent de mesurer une erreur et une vitesse est alors déterminée.

Les expérimentations effectuées sur la méthode proposée testent alors plusieurs facteurs que sont la robustesse vis à vis d'objets en mouvement, des variations des conditions de prise de vue ou la capacité à effectuer des boucles dans l'environnement. Pour ce qui est de la robustesse vis à vis des objets en mouvement l'algorithme développé dans le cadre de l'article est robuste tant qu'il peut suivre assez de points pour bien se situer, les points projetés sur des zones occultées étant rejetés car pas concluants. Pour tester les variations de prise de vue une navigation est effectuée sur une trajectoire acquise à un moment de la journée différente de celle qui a servi à créer le chemin visuel. Cela a pour effet de tester la transformation utilisée pour compenser les variations de luminosité entre les séquences. Bien que beaucoup de points soient correctement transformés certains sont assez mal mis en correspondance du fait des changements de luminance et c'est donc un problème potentiel de ce type d'approche. Enfin, pour ce qui est de la capacité à effectuer une boucle, différentes boucles ont été testées et ont monté que l'algorithme proposé s'adaptait bien.

3.1.3 Navigation par asservissement visuel direct

Les deux publications décrites plus haut présentent des techniques qui requièrent une extraction de points d'intérêt et une phase de traitement d'images avant de pouvoir effectuer un asservissement visuel du robot. Ceci peut être un problème car les primitives ne sont pas toujours triviales à extraire et des erreurs peuvent apparaître et avoir un effet néfaste sur la navigation. Pour essayer de palier ces problèmes potentiels et utiliser l'ensemble de l'information des images, les auteurs de [10] ont proposé une méthode de navigation reposant sur des aspects d'asservissement visuel direct et plus précisément utilisant une métrique d'information mutuelle afin de profiter des possibilités offertes par cette dernière.

Le même schéma que vu précédemment est utilisé : un robot suit une trajectoire et acquiert des images qui servent à créer un chemin visuel d'images clés puis lors de la navigation le robot détermine quelle image constitue sa cible locale et avance vers elle par asservissement visuel. Pour l'asservissement les auteurs s'appuient sur leurs précédents travaux ([8]) et définissent une commande pour le robot non-holonome basée sur l'information mutuelle. Une des difficultés de la méthode est la désignation des images clés. En effet comme aucune primitive n'est extraite il n'est pas possible de choisir les images avec un certain nombre de primitives en commun. L'information mutuelle étant sensée augmenter durant l'asservissement, il suffirait en théorie de choisir une nouvelle image quand

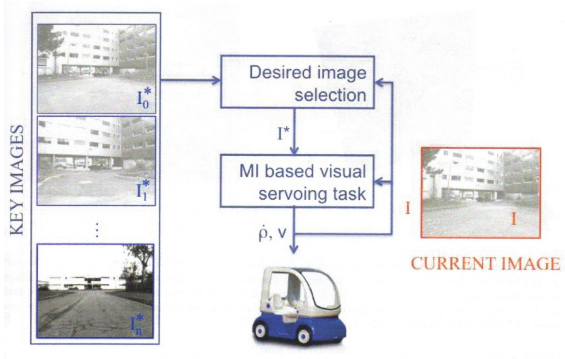


FIGURE 3.3 – Figure tirée de [10]. Boucle algorithmique de la navigation basée sur un asservissement visuel utilisant l'information mutuelle.

cette dernière commence à baisser mais cela ne marcherait que dans un monde parfait dans lequel les occlusions et autres perturbations du signal n'existent pas. Une solution alors proposée est de regarder la rotation appliqué au robot à chaque boucle. Lorsque cette dernière est très faible c'est que le robot est proche de la position désirée et qu'il peut passer à l'image suivante. Cependant cette solution pose des problèmes dans le cas d'une trajectoire en ligne droite. Pour remédier à cela l'incrément en rotation et celui en translation sont couplés afin d'éliminer ces problèmes.
Les avantages de cette méthode sont multiples. Premièrement aucune primitive n'est extraite ce qui fait qu'aucune erreur de tracking ou de matching n'est possible. L'approche considérée et son implémentation permettent une convergence rapide et ainsi un calcul à la cadence vidéo. Enfin l'asservissement visuel étant basé sur l'entropie des images il est résistant aux occlusions et aux variations de luminosité assez probables lors d'une navigation en extérieur.

3.1.4 Navigation par asservissement géométrique

Plusieurs travaux ont utilisé la vision catadioptrique pour réaliser des tâches de navigation. Par exemple, [25] l'utilise pour pour suivre des lignes au sol et se repérer dans des couloirs. Cet article présente une situation totalement différente que ce qui sera développé durant ce stage mais certains points peuvent être intéressants. Parmi ceux-ci figure l'utilisation des eigenspaces pour la représentation des images (utilisés également dans [26]). Les eigenspaces sont très utiles pour la représentation de l'environnement. En effet l'espace mémoire nécessaire pour stocker le monde sous forme d'images est très important et il est donc nécessaire de compresser ces informations. Les auteurs utilisent une représentation appelée *low dimensional eigenspace* décrite dans [18] qui est construite via une analyse en composantes principales, créant ainsi une base de vecteurs propres de basse dimension

représentant les images clés.
Dans [12] une approche proche de celle de [21] est utilisée. Le monde est représenté comme une carte composée d'images successives. Une extraction de points SIFT[16] et de“ segments colonnes” est effectuée et c'est en se basant sur ces informations que l'asservissement visuel est effectué.

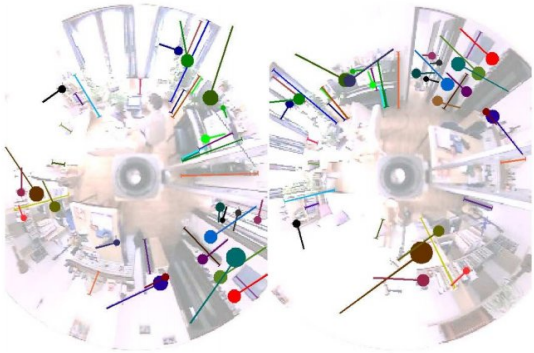


FIGURE 3.4 – Figure tirée de [12]. Mise en correspondance de points SIFT et de “segments colonnes”.

Ces méthodes utilisent des primitives extraites des images. Mais le problème de l'extraction est rendu encore plus complexe que sur des images perspectives du fait des transformations appliquées. Pour éviter cette source d'erreurs des méthodes directes ont également été proposées.

3.2 Description de la tâche de navigation

Une tâche de navigation ainsi définie est donc, par définition, une suite d'asservissements visuels. Cependant c'est la manière avec laquelle ces asservissements seront effectués qui est essentielle. En effet plusieurs questions se posent. Comment définir la trajectoire ? Quels degrés de liberté du robot contrôler ? Comment repérer le robot le long de la trajectoire ? Comment vérifier que le robot reste sur la trajectoire ? Ce sont tous ces paramètres qui déterminent la méthode de navigation et ce sont donc tous ces choix qui seront décrits dans ce chapitre.

3.3 Navigation par asservissement visuel entropique omnidirectionnel

Pour simplifier le contrôle du robot et éviter d'avoir à approcher une profondeur des points de la scène seul un degré de liberté en rotation sera piloté par asservissement visuel. Le robot avancera donc à une vitesse constante, ce qui évitera une série de décroissances exponentielles de la vitesse de déplacement et améliorera la continuité de mouvement du robot.

3.3.1 Chemin visuel

C'est en suivant un chemin visuel, comme défini par [21], que le robot effectuera la tâche de navigation. Le chemin à suivre est donc représenté par une suite d'images clés successives qui retracent ce que le robot voit lorsqu'il est baladé le long de sa trajectoire d'apprentissage. Pour déterminer ces images plusieurs solutions sont possibles. La solution choisie pour les tests est de capturer régulièrement des images à intervalle fixe pour avoir l'ensemble du chemin bien défini. Il ne s'agit pas d'une solution optimale. Les images pourraient être prises suivant leur dissimilarité afin d'assurer que les images sont pertinentes et éviter la redondance d'information. Cependant pour les première phases de tests ceci n'est pas critique et c'est le perfectionnement de la tâche de navigation qui a été privilégié. A l'heure où ce rapport est écrit une bonne partie du temps alloué pour le stage reste et ceci fait partie des améliorations envisagées qui seront peut-être apportées d'ici là.

3.3.2 Choix des images clés

Le point le plus important de la navigation utilisant un chemin visuel est de bien avancer le long de ce chemin. Le choix de changement d'image clé est donc critique. Étant donné que but est de maximiser une valeur d'information mutuelle l'idée a été de se tourner vers cette mesure. Il est envisageable que passé un certain seuil les images sont considérées comme assez proches et le robot peut donc avancer vers la prochaine. Cependant l'information mutuelle n'a pas de borne haute. Un seuil n'a donc dans ce contexte aucun sens, les ordres de grandeur pouvant changer d'une situation à une autre. Pour déterminer un moyen de passer outre ce problème c'est la définition de l'information mutuelle qui a servi de point de départ :

$$\mathrm{MI}(\mathbf{I}, \mathbf{I}^*) = \mathrm{H}(\mathbf{I}) + \mathrm{H}(\mathbf{I}^*) - \mathrm{H}(\mathbf{I}, \mathbf{I}^*). \tag{3.1}$$

Pour mieux se rendre compte voici une illustration de la situation sous la forme d'un diagramme de Venn :

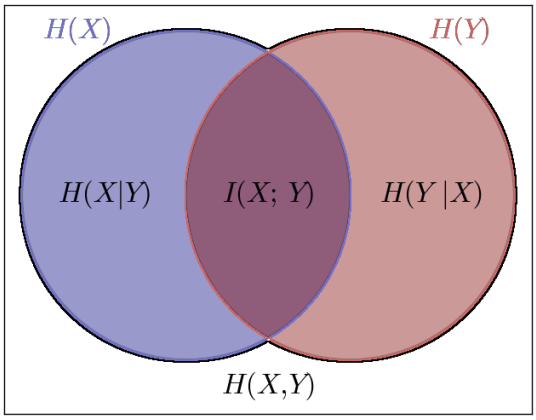


FIGURE 3.5 – Diagramme de Venn de l'information mutuelle. Tirée du site Wikipedia.

En extrapolant le modèle il apparaît que l'information mutuelle est comprise entre 0 et le minimum entre H($\mathbf{I}$) et H($\mathbf{I}^*$). En pratique, les images considérées (courante et référence) étant prises depuis des points de vue très proches, la différence d'entropie entre les deux est faible. C'est pour cela que le rapport de l'information mutuelle sur l'entropie de l'image clé courante a été choisi comme mesure de similarité. Ce choix a été fait pour garder un dénominateur constant sur chaque tâche d'asservissement. C'est à partir de cette mesure que le changement d'image clé va être déterminé. À chaque nouvelle acquisition d'image, la valeur d'information mutuelle est calculée et normalisée. Si cette dernière a diminué par rapport à l'image précédente c'est que le robot a dépassé le point de contrôle symbolisé par l'image clé. Une autre condition est de vérifier si la valeur d'information mutuelle stagne. Cela est effectué afin d'éviter de voir le robot finaliser l'asservissement visuel. Comme ce dernier va aller vers une autre image par la suite il n'est pas utile qu'il arrive exactement à la position de l'image clé, ce qui créerait une diminution de la vitesse à mesure qu'il s'en rapproche, et il est donc possible de changer d'image clé plus tôt afin de garder une vitesse plus constante. Pour limiter l'influence d'un bruit de mesure ces deux conditions sont assorties d'un seuil bas, ainsi si les deux images ne sont pas assez similaires le changement de référence est bloqué.

3.4 Résultats

La tâche de navigation utilisée pour les tests est simple. Après avoir acquis une trajectoire sous la forme d'une séquence d'images le robot est placé à la position de départ et

une vitesse de translation constante est appliquée à ce dernier (de préférence inférieure à la vitesse utilisée pour la capture afin de faciliter les convergences successives). L'asservissement visuel est alors effectué pour contrôler la vitesse de rotation. Le point critique de ces expériences est le réglage des différents paramètres de navigation qui ont été décrits plus haut. Il faut que le robot converge assez vite pour ne pas dépasser la position d'acquisition de l'image clé en cours mais il faut aussi qu'il effectue bien l'asservissement pour être au bon endroit. Changer trop tôt ou trop tard d'image clé peut fortement compromettre la tâche.

3.4.1 Simulateur

Avant de tester sur du vrai matériel il était préférable de vérifier que l'algorithme de navigation était correct. Pour cela un test a été effectué en simulation. Comme dans un premier temps la vérification de l'asservissement omnidirectionnel n'était pas nécessaire ceci a été fait sur des images avec un projection perpective. Pour avoir une simulation réaliste, le moteur de rendu 3D (Ogre3D) a été utilisé. Le modèle 3D d'un quartier de Paris sur lequel des textures réalistes ont été plaquées a été affiché et une caméra a été déplacée dans ce monde virtuel. Le rendu graphique enregistré a alors été utilisé comme chemin visuel. Le robot virtuel a ensuite été replacé au début de la trajectoire et la tâche a été lancée.

La courbe d'évolution de l'information mutuelle montre une chute brutale à chaque passage d'image clé puis remonte grâce à l'asservissement vers le nouveau point de passage. La simulation a permis de constater deux problèmes. Le premier est la sensibilité des seuils décidant du changement d'image clé et de la vitesse. En effet pour obtenir de bons résultats il faut que ces paramètres soient finement réglés. Le second est la vitesse envoyée au robot. Lors des changements de directions ou quand le passage d'image clé n'est pas bien réglé la vitesse calculée a tendance osciller autour de zéro. Ces oscillations semblent minimes mais pourront s'avérer gênantes pour un passager étant donné que pour le robot chaque changement de signe pour la vitesse correspond à un changement de direction. Ces oscillations conduisent donc à une instabilité du robot à son arrivée en ligne droite. Régler ceci fait partie des améliorations à apporter d'ici la fin du stage, notamment en passant par des solutions de filtrage.

3.4.2 Robot cartésien

Avant de valider le travail effectué avec un test grandeur nature un autre test a été fait sur le robot ayant servi pour l'asservissement visuel omnidirectionnel. Ce choix a été motivé par le fait que le robot utilisé pour les tests grandeur nature est complexe a utiliser et il vaut donc mieux avoir testé et validé l'algorithme à plus petite échelle avant de tester sur ce dernier. Cette étape a donc été insérée entre la simulation et le test grandeur réelle

Position courante

Image clé actuelle

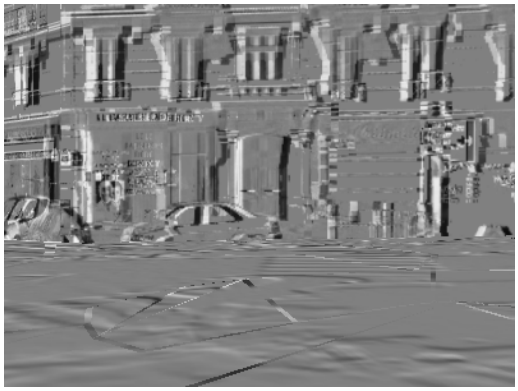
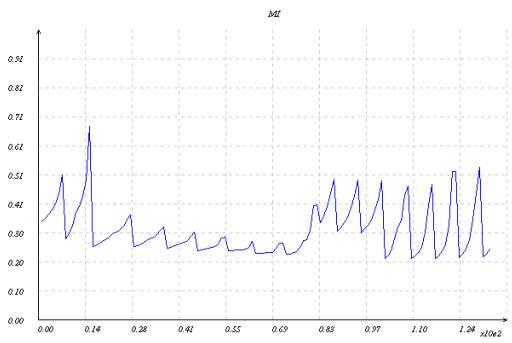


Image erreur

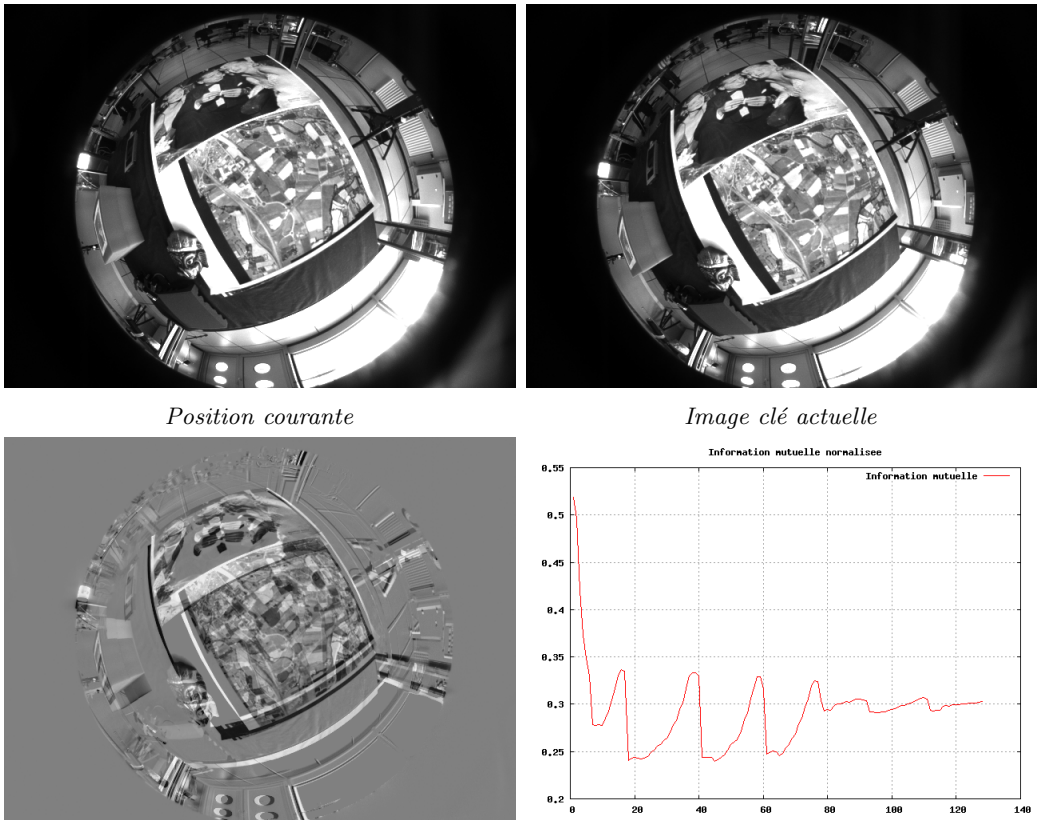


Évolution de l'information mutuelle.

FIGURE 3.6 – Données apportées par le simulateur de navigation.

afin de s'assurer de la viabilité de l'algorithme dans des situations réelles. Le protocole de test est très semblable à celui utilisé précédemment : le robot acquiert des images le long d'une trajectoire puis la navigation est lancée en se référant à ce chemin visuel. Ici encore seule une rotation est pilotée et une translation à vitesse constante est appliquée.

Ici aussi des croissances-décroissances de l'information mutuelle apparaîssent. Elles correspondent respectivement aux phase asservissements visuel et de changement d'image clé. En regardant l'image d'erreur il est possible de constater que les images sont décalées. Ceci est logique car le robot va moins vite dans ce cas que lors de l'acquisition des images mais cela n'affecte pas la tâche car la caméra omnidirectionnelle permet d'avoir beaucoup d'informations sur lesquelles se recaler. Les résultats diffèrent assez peu de ce qui a été obtenu en simulation. Cependant des problèmes potentiels se dessinent. Par exemple les propriétés de la scène influent sur les paramètres à choisir pour le passage des images clé. Le problème étant que ces dernières peuvent changer au cours de la navigation il serait donc



Position courante — *Image clé actuelle*

Image erreur — *Évolution de l'information mutuelle.*

FIGURE 3.7 – Données apportées par la simulation sur un robot cartésien.

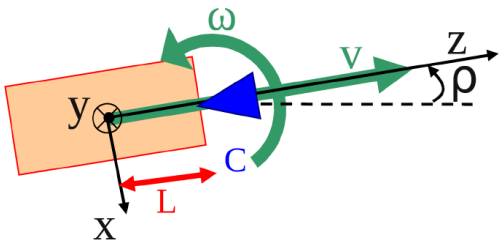
judicieux de réfléchir à un moyen de les ajuster ou d'adapter le système dynamiquement. Le temps de calcul commence aussi à devenir une composante critique. À chaque passage d'image clé la matrice hessienne de l'information mutuelle doit être remise à jour pour prendre en compte le template et la taille des images considérées implique un nombre assez important de calculs. Le problème est que pendant que ces calculs sont effectués le robot continue à avancer aux vitesses précédemment définies et un calcul trop long peut faire dévier le robot de la trajectoire et dans le pire des cas la perdre. Il va donc falloir accélérer les calculs. Dans cette optique, plusieurs solutions ont été envisagées. Il serait par exemple possible de tout calculer avant, au départ du robot, mais ceci demanderait beaucoup de temps et d'espace mémoire. Il serait également envisageable de les calculer en parallèle de la tâche d'asservissement étant donné que les machines utilisées sont multi-coeurs mais la complexité algorithmique introduite pourrait poser problème. Une autre solution serait de les calculer offline, par un autre programme en se basant sur le chemin

visuel et de les stocker dans une base de données. Mais pour que cela soit efficace il faudrait s'assurer que le temps de lecture de la matrice soit vraiment plus faible que son temps de calcul. Malheureusement à l'heure où ce rapport est écrit le choix d'une solution n'a pas encore été fait, ce problème ne s'étant posé que très récemment.

3.4.3 Robot non holonome

Malgré les légers problème potentiels soulevés par la validation sur le robot cartésien, le *framework* de navigation développé semble fonctionner et devrait donc être testé très prochainement sur un robot non holonome. Ce type de robot (voir figure 3.4.3) ne peut se déplacer comme un robot à six degrés de liberté. En effet, tout comme une voiture, il possède des contraintes qui limitent ses possibilités de mouvement. De plus, la vitesse appliquée à la caméra et celle appliquée à la voiture seront différentes vu la position de la caméra à l'avant du robot (voir figure ci-dessous). Enfin, le robot se déplaçant comme une voiture ce n'est pas une vitesse de rotation mais un angle à envoyer aux roues qu'il va falloir calculer.

Robot Cycab



Modèle considéré. Figure tirée de [7].

FIGURE 3.8 – Robot non holonome sur lequel doit être testée la navigation et le modèle associé.

En considérant le modèle il apparaît qu'il faudra transformer la vitesse calculée dans le repère de la caméra ω_y afin de l'envoyer aux roues. La formule donnant alors l'angle des roues ψ est :

$$\psi = arctan\left(\frac{L\omega_y}{v_z}\right) \tag{3.2}$$

Ces tests n'ont pas encore été effectués mais devrait l'être avant la fin du stage. En effet ils nécessitent beaucoup d'efforts logistiques et ne peuvent donc être réalisés que dans des conditions précises. Leur résultats permettront de valider la méthode complète et de prouver sa viabilité dans un environnement réel inconnu.

Conclusion

Dans ce chapitre, le *framework* de navigation utilisant l'asservissement visuel omnidirectionnel développé a été décrit. Les principales difficultés, notamment au niveau des changements d'image clé ont été expliqués. Des tests ont alors été décrits et leurs résultats interprétés. Malgré quelques problèmes, la possibilité de faire avancer un robot le long d'une trajectoire de manière autonome a été montrée. Enfin, un test grandeur nature qui reste à effectuer a été décrit.

Conclusion et perspectives

Durant cette première partie de stage de Master 2 une méthode de navigation autonome pour un robot utilisant un capteur catadioptrique a été imaginée et développée.

Pour cela les travaux utilisant l'information mutuelle ont servi de point de départ car cette méthode présente l'avantage d'utiliser un asservissement visuel direct et que l'information mutuelle présente des propriétés de robustesse très intéressantes pour une tâche de ce type. Comme ceci n'avait pas été étudié pour des images acquises avec des capteurs catadioptriques il a donc fallu recouper ces travaux avec d'autres travaux d'asservissement visuel omnidirectionnel. Ces études ont alors permis de déterminer un modèle permettant d'effectuer une commande de robot grâce à l'information mutuelle sur des capteurs omnidirectionnels. Ce modèle a alors été étendu afin de pouvoir effectuer une navigation autonome du robot se basant uniquement sur une séquence d'images.

Tout au long du stage les implémentations des différents modèles ont été validées par des expériences réalisées dans un premier temps sur un logiciel de simulation puis sur de vrais robots. Tous ces tests ont prouvé la validité de nos modèles, qu'il s'agisse de l'asservissement visuel omnidirectionnel basé sur l'information mutuelle ou la navigation basée sur cet asservissement, même si le véritable test grandeur nature ne pourra être réalisé que dans les semaines qui viennent.

Les expériences montrent que de nombreuses pistes existent pour améliorer les modèles. Pour ce qui est de l'asservissement d'autres représentations de la projection omnidirectionnelle existent et sont donc intéressantes à intégrer. Il est aussi possible de penser que la mesure de vraisemblance utilisée pour l'optimisation peut être revue, en particuliers il serait intéressant d'utiliser une métrique bornée. Enfin le processus d'optimisation en lui même et le lissage de la courbe de coût peuvent être améliorés. Pour ce qui est de la navigation, la méthode utilisée est très dépendante de paramètres choisis et il serait donc intéressant de réfléchir à des moyens de les adapter dynamiquement. Enfin il serait intéressant d'augmenter le nombre de degrés de liberté contrôlés par l'asservissement visuel, ce qui complexifierait le modèle mais donnerait plus de liberté et d'autonomie au robot.

Références

[1] G. Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Trans. on Pattern Analysis and Machine intelligence*, 7(4) :384–401, July 1985.

[2] H. Barreto, J.P. Araujo. Issues on the geometry of central catadioptric images. In *Int. Conf. on Computer Vision and Pattern Recognition*, Hawai, USA, December 2001.

[3] G. Caron, E. Marchand, and E. Mouaddib. Omnidirectional photometric visual servoing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'10*, pages 6202–6207, Taipei, Taiwan, October 2010.

[4] F. Chaumette and S. Hutchinson. Visual servo control, Part I : Basic approaches. *IEEE Robotics and Automation Magazine*, 13(4) :82–90, December 2006.

[5] C. Collewet and E. Marchand. Photometric visual servoing. *IEEE Int. Trans. on Robotics and Automation*, PP(99) :1–7, 2011.

[6] C. Collewet, E. Marchand, and F. Chaumette. Visual servoing set free from image processing. In *IEEE Int. Conf. on Robotics and Automation, ICRA'08*, Pasadena, CA, May 2008.

[7] A. Dame. *A unified direct approach for visual servoing and visual tracking using mutual information*. PhD thesis, Université de Rennes1, Rennes, France, December 2010.

[8] A. Dame and E. Marchand. Entropy based visual servoing. In *IEEE Int. Conf. on Robotics and Automation, ICRA'09*, pages 707–713, Kobe, Japan, May 2009.

[9] A. Dame and E. Marchand. Improving mutual information based visual servoing. In *IEEE Int. Conf. on Robotics and Automation, ICRA'10*, pages 5531–5536, Anchorage, Alaska, May 2010.

[10] A. Dame and E. Marchand. A new information theoretic approach for apparence-based visual path following. In *IEEE Int. Conf. on Robotics and Automation, ICRA'11*, Shangaï, China, May 2011.

[11] N. Fischler and R.C. Bolles. Random sample consensus : A paradigm for model fitting with application to image analysis and automated cartography. *Communication of the ACM*, 24(6) :381–395, June 1981.

[12] Toon Goedemé, Tinne Tuytelaars, and Luc Van Gool. Omnidirectional sparse visual path following with occlusion-robust feature tracking. In *in : 6th Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras, OMNIVIS05, in Conjunction with ICCV 2005*, 2005.

[13] H. Hadj-Abdelkader. *Asservissement visuel en vision omni-directionnelle*. PhD thesis, Université Blaise Pascal , Clermont II, Clermont, France, November 2006.

[14] R. Hamel, T. Mahony. Visual servoing of an under-actuated dynamic rigid-body system : An image-based approach. *IEEE Int. Trans. on Robotics and Automation*, Vol. 18, No. 2, 2002.

[15] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Conference*, pages 147–151, Manchester, 1988.

[16] D. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 60(2) :91–110, 2004.

[17] E. Mouaddib. Introduction à la vision panoramique catadioptrique. In *Traitement du signal, vol.22, n.5 Vision Omnidirectionnelle, 2005*, Amiens, France, Septembre 2005.

[18] H. Murase and S.K. Nayar. Visual learning and recognition of 3d objects from appearance. *Int. J. of Computer Vision*, 14 :5–24, 1995.

[19] J.P.W. Pluim, J.B.A. Maintz, and M.A. Viergever. Mutual information matching and interpolation artefacts. In K.M. Hanson, editor, *SPIE Medical Imaging*, volume 3661, pages 56–65. SPIE Press, 1999.

[20] E. Royer, M. Lhuillier, M. Dhome, and J.M. Lavest. Monocular vision for mobile robot localization and autonomous navigation. *International Journal of Computer Vision*, 74(3) :237–260, 2007.

[21] S. Segvic, A. Remazeilles, A. Diosi, and F. Chaumette. A mapping and localization framework for scalable appearance-based navigation. *Computer Vision and Image Understanding*, 113(2) :172–187, February 2009.

[22] CE. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27, 1948.

[23] D. Studholme, C. Hill and D. Hawkes. An overlap invariant entropy mesure of 3d medical image alignment. *Pattern Recognition*, 32(99) :71–86, 1999.

[24] P. Thévenaz and M. Unser. Optimization of Mutual Information for Multiresolution Image Registration. *IEEE trans. on Image Processing*, 9(12) :2083–2099, 2000.

[25] N. Winters, J. Gaspar, G. Lacey, and J. Santos-Victor. Omni-directional vision for robot navigation. *IEEE Workshop on Omnidirectional Vision, OMNIVIS'00*, page 21, 2000.

[26] Niall Winters and Jose Santos-victor. Mobile robot navigation using omni-directional vision. In *in 3rd Irish Machine Vision and Image Processing Conference, IMVIP'99*, pages 151–166, 1999.

[27] Xianghua Ying and Zhanyi Hu. Can We Consider Central Catadioptric Cameras and Fisheye Cameras within a Unified Imaging Model ? In *European Conference on Computer Vision*, volume 1, Prague, Czech, 2004.

Résumé

Dans ce rapport sont traités les problèmes d'asservissement visuel et de navigation, des sujets très actifs dans le domaine de la vision robotique. Beaucoup de travaux existent, principalement utilisant des techniques dites géométriques, c'est à dire se basant sur des primitives géométriques. Ces techniques nécessitant une phase de suivi ou d'extraction elles n'utilisent pas toute l'information de l'image. C'est pour cela qu'ont été développées récemment des techniques, dites directes, qui utilisent toute l'image sans suivre ou extraire de primitives géométriques. Une de ces techniques se base sur l'information mutuelle ([22]) des images et a été développée pour des images acquises par des caméras *perspective* ([7]). Cette méthode a pour principal avantage le fait de posséder les propriétés de robustesse dues à l'information mutuelle, notamment vis à vis des changements d'illumination, des occlusions ou de la multimodalité. La limite de cette approche réside dans le fait que des images de ce type amènent peu d'information du fait de leur champ de vision réduit.

Pour dépasser cette limite, une solution possible est de l'adapter à des images omnidirectionnelles qui apportent une information sur la scène à 360 $^\circ$. Dans ce rapport seront rappelés dans un premier temps les travaux existants dans le domaine de l'asservissement visuel avec des caméras perspective. Dans un second temps les méthodes d'asservissement visuel omnidirectionnel serons introduites avant que notre méthode soit développée. Cette dernière sera par la suite validée par des tests expérimentaux réalisés sur un robot cartésien à six degrés de liberté. Enfin dans une troisième partie un rappel des travaux sur la navigation de robot sera fait avant que ne soit montré comment cet asservissement peut être adapté à la navigation. La méthode proposée sera ici aussi validée avec cette fois des expériences successives en simulation, sur le robot cartésien à six degrés de liberté avant que ne soit détaillé un test grandeur nature sur un robot mobile non holonome.